Lab 3

In [40]:

```
# install libraries
!pip install tensorflow==2.6.0
!pip install keras==2.6.0
!pip install matplotlib==3.2.2
!pip install numpy==1.22.0
!pip install seaborn==0.11.1
!pip install scikit-learn==0.23.1
!pip install opencv-python
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: tensorflow==2.6.0 in /home/zz3904/.local/lib/pyth
on3.8/site-packages (2.6.0)
Requirement already satisfied: flatbuffers~=1.12.0 in /home/zz3904/.local/lib/py
thon3.8/site-packages (from tensorflow==2.6.0) (1.12)
Requirement already satisfied: keras~=2.6 in /home/zz3904/.local/lib/python3.8/s
ite-packages (from tensorflow==2.6.0) (2.6.0)
Requirement already satisfied: wrapt~=1.12.1 in /home/zz3904/.local/lib/python3.
8/site-packages (from tensorflow==2.6.0) (1.12.1)
Requirement already satisfied: opt-einsum~=3.3.0 in /home/zz3904/.local/lib/pyth
on3.8/site-packages (from tensorflow==2.6.0) (3.3.0)
Requirement already satisfied: keras-preprocessing~=1.1.2 in /home/zz3904/.loca
l/lib/python3.8/site-packages (from tensorflow==2.6.0) (1.1.2)
Requirement already satisfied: gast==0.4.0 in /home/zz3904/.local/lib/python3.8/
site-packages (from tensorflow==2.6.0) (0.4.0)
Requirement already satisfied: termcolor~=1.1.0 in /home/zz3904/.local/lib/pytho
n3.8/site-packages (from tensorflow==2.6.0) (1.1.0)
Requirement already satisfied: tensorflow-estimator~=2.6 in /home/zz3904/.local/
lib/python3.8/site-packages (from tensorflow==2.6.0) (2.8.0)
Requirement already satisfied: six~=1.15.0 in /share/apps/python/3.8.6/intel/li
b/python3.8/site-packages (from tensorflow==2.6.0) (1.15.0)
Requirement already satisfied: astunparse~=1.6.3 in /home/zz3904/.local/lib/pyth
on3.8/site-packages (from tensorflow==2.6.0) (1.6.3)
Collecting numpy~=1.19.2
  Using cached numpy-1.19.5-cp38-cp38-manylinux2010_x86_64.whl (14.9 MB)
Requirement already satisfied: grpcio<2.0,>=1.37.0 in /home/zz3904/.local/lib/py
thon3.8/site-packages (from tensorflow==2.6.0) (1.44.0)
Requirement already satisfied: wheel~=0.35 in /share/apps/python/3.8.6/intel/li
b/python3.8/site-packages (from tensorflow==2.6.0) (0.35.1)
Requirement already satisfied: typing-extensions~=3.7.4 in /home/zz3904/.local/l
ib/python3.8/site-packages (from tensorflow==2.6.0) (3.7.4.3)
Requirement already satisfied: clang~=5.0 in /home/zz3904/.local/lib/python3.8/s
ite-packages (from tensorflow==2.6.0) (5.0)
Requirement already satisfied: tensorboard~=2.6 in /home/zz3904/.local/lib/pytho
n3.8/site-packages (from tensorflow==2.6.0) (2.8.0)
Requirement already satisfied: h5py~=3.1.0 in /home/zz3904/.local/lib/python3.8/
site-packages (from tensorflow==2.6.0) (3.1.0)
Requirement already satisfied: absl-py~=0.10 in /share/apps/python/3.8.6/intel/l
ib/python3.8/site-packages (from tensorflow==2.6.0) (0.13.0)
Requirement already satisfied: protobuf>=3.9.2 in /home/zz3904/.local/lib/python
3.8/site-packages (from tensorflow==2.6.0) (3.19.4)
Requirement already satisfied: google-pasta~=0.2 in /home/zz3904/.local/lib/pyth
on3.8/site-packages (from tensorflow==2.6.0) (0.2.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /home/zz3904/.lo
cal/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow==2.6.0) (1.8.
1)
Requirement already satisfied: setuptools>=41.0.0 in /share/apps/python/3.8.6/in
tel/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow==2.6.0) (49.
2.1)
Requirement already satisfied: werkzeug>=0.11.15 in /home/zz3904/.local/lib/pyth
on3.8/site-packages (from tensorboard~=2.6->tensorflow==2.6.0) (2.0.3)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /home/zz390
4/.local/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow==2.6.0)
(0.4.6)
Requirement already satisfied: markdown>=2.6.8 in /home/zz3904/.local/lib/python
3.8/site-packages (from tensorboard~=2.6->tensorflow==2.6.0) (3.3.6)
Requirement already satisfied: google-auth<3,>=1.6.3 in /home/zz3904/.local/lib/
python3.8/site-packages (from tensorboard~=2.6->tensorflow==2.6.0) (2.6.2)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /home/zz
```

```
3904/.local/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow==2.6.
0) (0.6.1)
Requirement already satisfied: requests<3,>=2.21.0 in /share/apps/python/3.8.6/i
ntel/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow==2.6.0) (2.2
4.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /home/zz3904/.local/l
ib/python3.8/site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=
2.6->tensorflow==2.6.0) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4; python_version < "3.10"
in /home/zz3904/.local/lib/python3.8/site-packages (from markdown>=2.6.8->tensor
board~=2.6->tensorflow==2.6.0) (4.11.3)
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3.6" in /home/z
z3904/.local/lib/python3.8/site-packages (from google-auth<3,>=1.6.3->tensorboar
d~=2.6->tensorflow==2.6.0) (4.8)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /home/zz3904/.local/li
b/python3.8/site-packages (from google-auth<3,>=1.6.3->tensorboard~=2.6->tensorf
low==2.6.0) (5.0.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /home/zz3904/.local/lib/
python3.8/site-packages (from google-auth<3,>=1.6.3->tensorboard~=2.6->tensorflo
w==2.6.0) (0.2.8)
Requirement already satisfied: idna<3,>=2.5 in /share/apps/python/3.8.6/intel/li
b/python3.8/site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflo
w==2.6.0) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /share/apps/python/3.8.6/int
el/lib/python3.8/site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tens
orflow==2.6.0) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /share/apps/python/3.8.6/in
tel/lib/python3.8/site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->ten
sorflow==2.6.0) (2020.6.20)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /shar
e/apps/python/3.8.6/intel/lib/python3.8/site-packages (from requests<3,>=2.21.0-
>tensorboard~=2.6->tensorflow==2.6.0) (1.25.10)
Requirement already satisfied: oauthlib>=3.0.0 in /share/apps/python/3.8.6/inte
l/lib/python3.8/site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthl
ib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow==2.6.0) (3.1.0)
Requirement already satisfied: zipp>=0.5 in /home/zz3904/.local/lib/python3.8/si
te-packages (from importlib-metadata>=4.4; python_version < "3.10"->markdown>=2.
6.8->tensorboard~=2.6->tensorflow==2.6.0) (3.7.0)
Requirement already satisfied: pyasn1>=0.1.3 in /home/zz3904/.local/lib/python3.
8/site-packages (from rsa<5,>=3.1.4; python_version >= "3.6"->google-auth<3,>=1.
6.3->tensorboard~=2.6->tensorflow==2.6.0) (0.4.8)
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.22.0
    Uninstalling numpy-1.22.0:
      Successfully uninstalled numpy-1.22.0
Successfully installed numpy-1.19.5
WARNING: You are using pip version 20.2.3; however, version 22.0.4 is available.
You should consider upgrading via the '/share/apps/python/3.8.6/intel/bin/python
-m pip install --upgrade pip' command.
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: keras==2.6.0 in /home/zz3904/.local/lib/python3.
8/site-packages (2.6.0)
WARNING: You are using pip version 20.2.3; however, version 22.0.4 is available.
You should consider upgrading via the '/share/apps/python/3.8.6/intel/bin/python
-m pip install --upgrade pip' command.
Defaulting to user installation because normal site-packages is not writeable
Collecting matplotlib==3.2.2
  Using cached matplotlib-3.2.2-cp38-cp38-manylinux1_x86_64.whl (12.4 MB)
```

```
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /shar
e/apps/python/3.8.6/intel/lib/python3.8/site-packages (from matplotlib==3.2.2)
 (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /share/apps/python/3.8.6/int
el/lib/python3.8/site-packages (from matplotlib==3.2.2) (1.2.0)
Requirement already satisfied: numpy>=1.11 in /home/zz3904/.local/lib/python3.8/
site-packages (from matplotlib==3.2.2) (1.19.5)
Requirement already satisfied: python-dateutil>=2.1 in /share/apps/python/3.8.6/
intel/lib/python3.8/site-packages (from matplotlib==3.2.2) (2.8.1)
Requirement already satisfied: cycler>=0.10 in /share/apps/python/3.8.6/intel/li
b/python3.8/site-packages (from matplotlib==3.2.2) (0.10.0)
Requirement already satisfied: six>=1.5 in /share/apps/python/3.8.6/intel/lib/py
thon3.8/site-packages (from python-dateutil>=2.1->matplotlib==3.2.2) (1.15.0)
Installing collected packages: matplotlib
Successfully installed matplotlib-3.2.2
WARNING: You are using pip version 20.2.3; however, version 22.0.4 is available.
You should consider upgrading via the '/share/apps/python/3.8.6/intel/bin/python
-m pip install --upgrade pip' command.
Defaulting to user installation because normal site-packages is not writeable
Collecting numpy==1.22.0
  Using cached numpy-1.22.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (16.8 MB)
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.19.5
    Uninstalling numpy-1.19.5:
      Successfully uninstalled numpy-1.19.5
ERROR: After October 2020 you may experience errors when installing or updating
 packages. This is because pip will change the way that it resolves dependency c
onflicts.

We recommend you use --use-feature=2020-resolver to test your packages with the
 new resolver before it becomes the default.

tensorflow 2.6.0 requires numpy~=1.19.2, but you'll have numpy 1.22.0 which is i
ncompatible.
tensorflow-gpu 2.6.0 requires numpy~=1.19.2, but you'll have numpy 1.22.0 which
 is incompatible.
Successfully installed numpy-1.22.0
WARNING: You are using pip version 20.2.3; however, version 22.0.4 is available.
You should consider upgrading via the '/share/apps/python/3.8.6/intel/bin/python
-m pip install --upgrade pip' command.
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: seaborn==0.11.1 in /home/zz3904/.local/lib/python
3.8/site-packages (0.11.1)
Requirement already satisfied: numpy>=1.15 in /home/zz3904/.local/lib/python3.8/
site-packages (from seaborn==0.11.1) (1.22.0)
Requirement already satisfied: pandas>=0.23 in /share/apps/python/3.8.6/intel/li
b/python3.8/site-packages (from seaborn==0.11.1) (1.1.3)
Requirement already satisfied: scipy>=1.0 in /share/apps/python/3.8.6/intel/lib/
python3.8/site-packages/scipy-1.5.2-py3.8-linux-x86_64.egg (from seaborn==0.11.
1) (1.5.2)
Requirement already satisfied: matplotlib>=2.2 in /share/apps/python/3.8.6/inte
l/lib/python3.8/site-packages (from seaborn==0.11.1) (3.3.2)
Requirement already satisfied: python-dateutil>=2.7.3 in /share/apps/python/3.8.
6/intel/lib/python3.8/site-packages (from pandas>=0.23->seaborn==0.11.1) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /share/apps/python/3.8.6/intel/li
b/python3.8/site-packages (from pandas>=0.23->seaborn==0.11.1) (2020.1)
Requirement already satisfied: certifi>=2020.06.20 in /share/apps/python/3.8.6/i
```

```
ntel/lib/python3.8/site-packages (from matplotlib>=2.2->seaborn==0.11.1) (2020.
6.20)
Requirement already satisfied: cycler>=0.10 in /share/apps/python/3.8.6/intel/li
b/python3.8/site-packages (from matplotlib>=2.2->seaborn==0.11.1) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in /share/apps/python/3.8.6/intel/l
ib/python3.8/site-packages (from matplotlib>=2.2->seaborn==0.11.1) (8.0.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /shar
e/apps/python/3.8.6/intel/lib/python3.8/site-packages (from matplotlib>=2.2->sea
born==0.11.1) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /share/apps/python/3.8.6/int
el/lib/python3.8/site-packages (from matplotlib>=2.2->seaborn==0.11.1) (1.2.0)
Requirement already satisfied: six>=1.5 in /share/apps/python/3.8.6/intel/lib/py
thon3.8/site-packages (from python-dateutil>=2.7.3->pandas>=0.23->seaborn==0.11.
1) (1.15.0)
WARNING: You are using pip version 20.2.3; however, version 22.0.4 is available.
You should consider upgrading via the '/share/apps/python/3.8.6/intel/bin/python
-m pip install --upgrade pip' command.
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scikit-learn==0.23.1 in /home/zz3904/.local/lib/p
ython3.8/site-packages (0.23.1)
Requirement already satisfied: scipy>=0.19.1 in /share/apps/python/3.8.6/intel/l
ib/python3.8/site-packages/scipy-1.5.2-py3.8-linux-x86_64.egg (from scikit-learn
==0.23.1) (1.5.2)
Requirement already satisfied: joblib>=0.11 in /home/zz3904/.local/lib/python3.
8/site-packages (from scikit-learn==0.23.1) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /home/zz3904/.local/lib/p
ython3.8/site-packages (from scikit-learn==0.23.1) (3.1.0)
Requirement already satisfied: numpy>=1.13.3 in /home/zz3904/.local/lib/python3.
8/site-packages (from scikit-learn==0.23.1) (1.22.0)
WARNING: You are using pip version 20.2.3; however, version 22.0.4 is available.
You should consider upgrading via the '/share/apps/python/3.8.6/intel/bin/python
-m pip install --upgrade pip' command.
Defaulting to user installation because normal site-packages is not writeable
Collecting opencv-python
  Downloading opencv_python-4.5.5.64-cp36-abi3-manylinux_2_17_x86_64.manylinux20
14_x86_64.whl (60.5 MB)
     |████████████████████████████████| 60.5 MB 22.9 MB/s eta 0:00:01
Requirement already satisfied: numpy>=1.17.3; python_version >= "3.8" in /home/z
z3904/.local/lib/python3.8/site-packages (from opencv-python) (1.22.0)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.5.5.64
WARNING: You are using pip version 20.2.3; however, version 22.0.4 is available.
You should consider upgrading via the '/share/apps/python/3.8.6/intel/bin/python
-m pip install --upgrade pip' command.
```

In [2]:
```python
# use GPU
import tensorflow as tf
import keras
from tensorflow.compat.v1.keras.backend import set_session
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
/home/zz3904/.local/lib/python3.8/site-packages/numpy/core/getlimits.py:499: UserWarning: The value of the smallest subnormal for <class 'numpy.float64'> type is zero.
  setattr(self, word, getattr(machar, word).flat[0])
/home/zz3904/.local/lib/python3.8/site-packages/numpy/core/getlimits.py:89: UserWarning: The value of the smallest subnormal for <class 'numpy.float64'> type is zero.
  return self._float_to_str(self.smallest_subnormal)
/home/zz3904/.local/lib/python3.8/site-packages/numpy/core/getlimits.py:499: UserWarning: The value of the smallest subnormal for <class 'numpy.float32'> type is zero.
  setattr(self, word, getattr(machar, word).flat[0])
/home/zz3904/.local/lib/python3.8/site-packages/numpy/core/getlimits.py:89: UserWarning: The value of the smallest subnormal for <class 'numpy.float32'> type is zero.
  return self._float_to_str(self.smallest_subnormal)

[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 5639310990961501112
]
```

In [3]:
```python
config = tf.compat.v1.ConfigProto( device_count = {'GPU': 1 , 'CPU': 56} )
sess = tf.compat.v1.Session(config=config)
set_session(sess)
```

Problem 1 - Weight Initialization, Dead Neurons, Leaky ReLU

Part 1

Explain vanishing gradients phenomenon using standard normalization with different values of standard deviation and tanh and sigmoid activation functions. Then show how Xavier (aka Glorot normal) initialization of weights helps in dealing with this problem. Next use ReLU activation and show that instead of Xavier initialization, He initialization works better for ReLU activation. You can plot activations at each of the 5 layers to answer this question. (8)

Written Answer

- When many layers of neural network using certain activation functions like sigmoid, the gradients of the activation function are close to 0, which causes the model more difficult to train.
- Tanh and sigmoid action functions convert many inputs into a small number of inputs from 0 and 1. A significant move in the input of the sigmoid activation function only results in a small change in the output, which means the derivative is small.

How Xavier initialization of weights helps in dealing with this problem

- Xavier initialization tries to keep all the “winning features listed, that is, gradients, Z-values and Activations similar along all the layers. Another way of putting it: keeping variance similar along all the layers.”!

He initialization works better for ReLU activation.

- “the ReLU turns half of the Z-values (the negative ones) into zeros, effectively removing about half of the variance. So, we need to double the variance of the weights to compensate for it.”

Code

Utility functions to create a sample model and for creating graphs

- From Andre Perunicic. Understand neural network weight initialization.

In [6]:

```python
# 1
import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
from keras import backend as K

from matplotlib import pyplot as plt
from matplotlib import rcParamsDefault
from tensorflow.keras import optimizers

def grid_axes_it(n_plots, n_cols=3, enumerate=False, fig=None):
    """
    Iterate through Axes objects on a grid with n_cols columns and as many
    rows as needed to accommodate n_plots many plots.
    Args:
        n_plots: Number of plots to plot onto figure.
        n_cols: Number of columns to divide the figure into.
        fig: Optional figure reference.
    Yields:
        n_plots many Axes objects on a grid.
    """
    n_rows = n_plots / n_cols + int(n_plots % n_cols > 0)

    if not fig:
        default_figsize = rcParamsDefault['figure.figsize']
        fig = plt.figure(figsize=(
            default_figsize[0] * n_cols,
            default_figsize[1] * n_rows
        ))

    for i in range(1, n_plots + 1):
        ax = plt.subplot(n_rows, n_cols, i)
        yield ax


def create_mlp_model(
    n_hidden_layers,
    dim_layer,
    input_shape,
    n_classes,
    kernel_initializer,
    bias_initializer,
    activation,
):
    """Create Multi-Layer Perceptron with given parameters."""
    model = Sequential()
    model.add(Dense(dim_layer, input_shape=input_shape, kernel_initializer=kernel
_initializer,
                    bias_initializer=bias_initializer))
    for i in range(n_hidden_layers):
        model.add(Dense(dim_layer, activation=activation, kernel_initializer=kern
el_initializer,
                        bias_initializer=bias_initializer))
    model.add(Dense(n_classes, activation='softmax', kernel_initializer=kernel_in
itializer,
                    bias_initializer=bias_initializer))
    return model
```

```
def create_cnn_model(input_shape, num_classes, kernel_initializer='glorot_unifor
m',
                     bias_initializer='zeros'):
    """Create CNN model similar to
       https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py."""
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3),
                     activation='relu',
                     input_shape=input_shape,
                     kernel_initializer=kernel_initializer,
                     bias_initializer=bias_initializer))
    model.add(Conv2D(64, (3, 3), activation='relu',
                     kernel_initializer=kernel_initializer,
                     bias_initializer=bias_initializer))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu',
                    kernel_initializer=kernel_initializer,
                    bias_initializer=bias_initializer))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax',
                    kernel_initializer=kernel_initializer,
                    bias_initializer=bias_initializer))
    return model


def compile_model(model):
    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=optimizers.RMSprop(),
                  metrics=['accuracy'])
    return model


def get_init_id(init):
    """
    Returns string ID summarizing initialization scheme and its parameters.
    Args:
        init: Instance of some initializer from keras.initializers.
    """
    try:
        init_name = str(init).split('.')[2].split(' ')[0]
    except:
        init_name = str(init).split(' ')[0].replace('.', '_')

    param_list = []
    config = init.get_config()
    for k, v in config.items():
        if k == 'seed':
            continue
        param_list.append('{k}-{v}'.format(k=k, v=v))
    init_params = '__'.join(param_list)

    return '|'.join([init_name, init_params])


def get_activations(model, x, mode=0.0):
    """Extract activations with given model and input vector x."""
    outputs = [layer.output for layer in model.layers]
```

```
    activations = K.function([model.input], outputs)
    output_elts = activations(x)
    return output_elts


class LossHistory(keras.callbacks.Callback):
    """A custom keras callback for recording losses during network training."""

    def on_train_begin(self, logs={}):
        self.losses = []
        self.epoch_losses = []
        self.epoch_val_losses = []

    def on_batch_end(self, batch, logs={}):
        self.losses.append(logs.get('loss'))

    def on_epoch_end(self, epoch, logs={}):
        self.epoch_losses.append(logs.get('loss'))
        self.epoch_val_losses.append(logs.get('val_loss'))
```

Plot activation layers

In [7]:
```python
import keras
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from keras import initializers
from keras.datasets import mnist

seed = 10

# Number of points to plot
n_train = 1000
n_test = 100
n_classes = 10

# Network params
n_hidden_layers = 5
dim_layer = 100
batch_size = n_train
epochs = 1

# Load and prepare MNIST dataset.
n_train = 60000
n_test = 10000

(x_train, y_train), (x_test, y_test) = mnist.load_data()
num_classes = len(np.unique(y_test))
data_dim = 28 * 28

x_train = x_train.reshape(60000, 784).astype('float32')[:n_train]
x_test = x_test.reshape(10000, 784).astype('float32')[:n_train]
x_train /= 255
x_test /= 255

y_train = keras.utils.np_utils.to_categorical(y_train, num_classes)
y_test = keras.utils.np_utils.to_categorical(y_test, num_classes)

# Run the data through a few MLP models and save the activations from
# each layer into a Pandas DataFrame.
rows = []
sigmas = [0.10, 0.14, 0.28]
for stddev in sigmas:
    init = initializers.RandomNormal(mean=0.0, stddev=stddev, seed=seed)
    activation = 'tanh'

    model = create_mlp_model(
        n_hidden_layers,
        dim_layer,
        (data_dim,),
        n_classes,
        init,
        'zeros',
        activation
    )
    compile_model(model)
    output_elts = get_activations(model, x_test)
    n_layers = len(model.layers)
    i_output_layer = n_layers - 1
```

```
    for i, out in enumerate(output_elts[:-1]):
        if i > 0 and i != i_output_layer:
            for out_i in out.ravel()[::20]:
                rows.append([i, stddev, out_i])

df = pd.DataFrame(rows, columns=['Hidden Layer', 'Standard Deviation', 'Output'])

# Plot previously saved activations from the 5 hidden layers
# using different initialization schemes.
fig = plt.figure(figsize=(12, 6))
axes = grid_axes_it(len(sigmas), 1, fig=fig)
for sig in sigmas:
    ax = next(axes)
    ddf = df[df['Standard Deviation'] == sig]
    sns.violinplot(x='Hidden Layer', y='Output', data=ddf, ax=ax, scale='count',
inner=None)

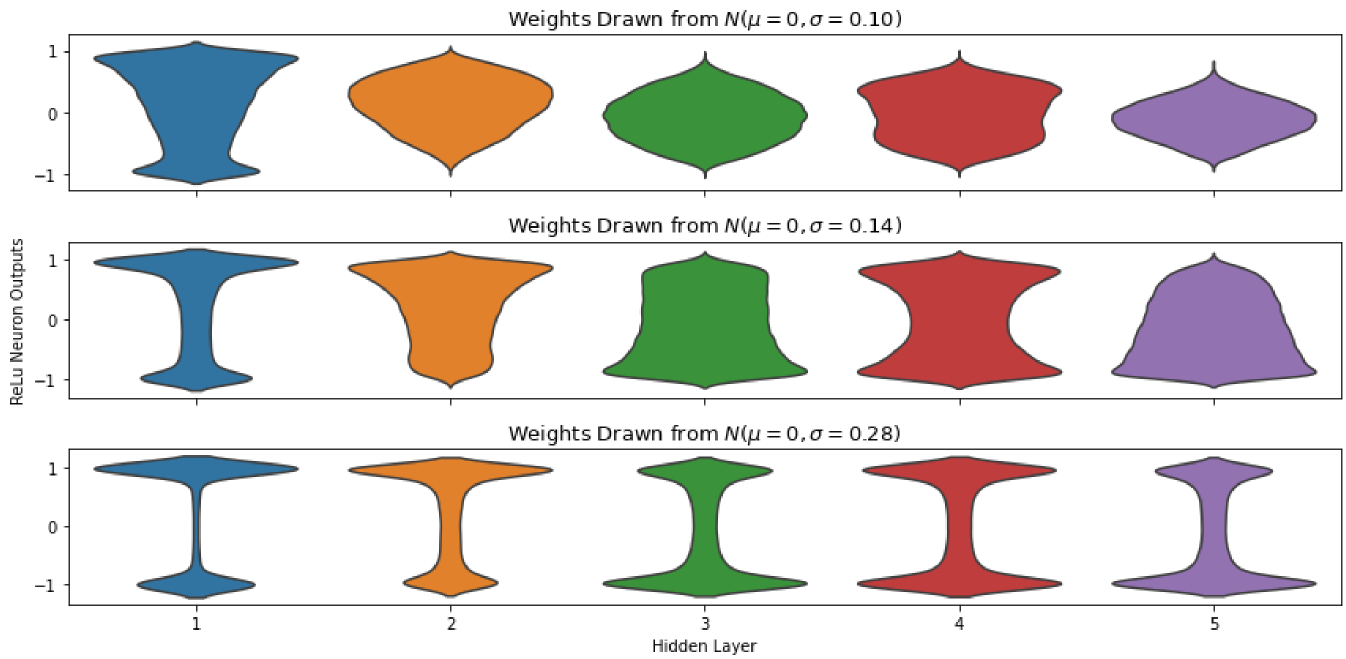
    ax.set_xlabel('')
    ax.set_ylabel('')

    ax.set_title('Weights Drawn from $N(\mu = 0, \sigma = {%.2f})$' % sig, fontsi
ze=13)

    if sig == sigmas[1]:
        ax.set_ylabel("ReLu Neuron Outputs")
    if sig != sigmas[-1]:
        ax.set_xticklabels(())
    else:
        ax.set_xlabel("Hidden Layer")

plt.tight_layout()
plt.show()
```

```
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
```



Use Sigmoid activation function

In [8]:

```
sigmas = [0.10, 0.14, 0.28]
for stddev in sigmas:
    init = initializers.RandomNormal(mean=0.0, stddev=stddev, seed=seed)
    activation = 'sigmoid'

    model = create_mlp_model(
        n_hidden_layers,
        dim_layer,
        (data_dim,),
        n_classes,
        init,
        'zeros',
        activation
    )
    compile_model(model)
    output_elts = get_activations(model, x_test)
    n_layers = len(model.layers)
    i_output_layer = n_layers - 1

    for i, out in enumerate(output_elts[:-1]):
        if i > 0 and i != i_output_layer:
            for out_i in out.ravel()[::20]:
                rows.append([i, stddev, out_i])

df = pd.DataFrame(rows, columns=['Hidden Layer', 'Standard Deviation', 'Output'])

# Plot previously saved activations from the 5 hidden layers
# using different initialization schemes.
fig = plt.figure(figsize=(12, 6))
axes = grid_axes_it(len(sigmas), 1, fig=fig)
for sig in sigmas:
    ax = next(axes)
    ddf = df[df['Standard Deviation'] == sig]
    sns.violinplot(x='Hidden Layer', y='Output', data=ddf, ax=ax, scale='count',
inner=None)

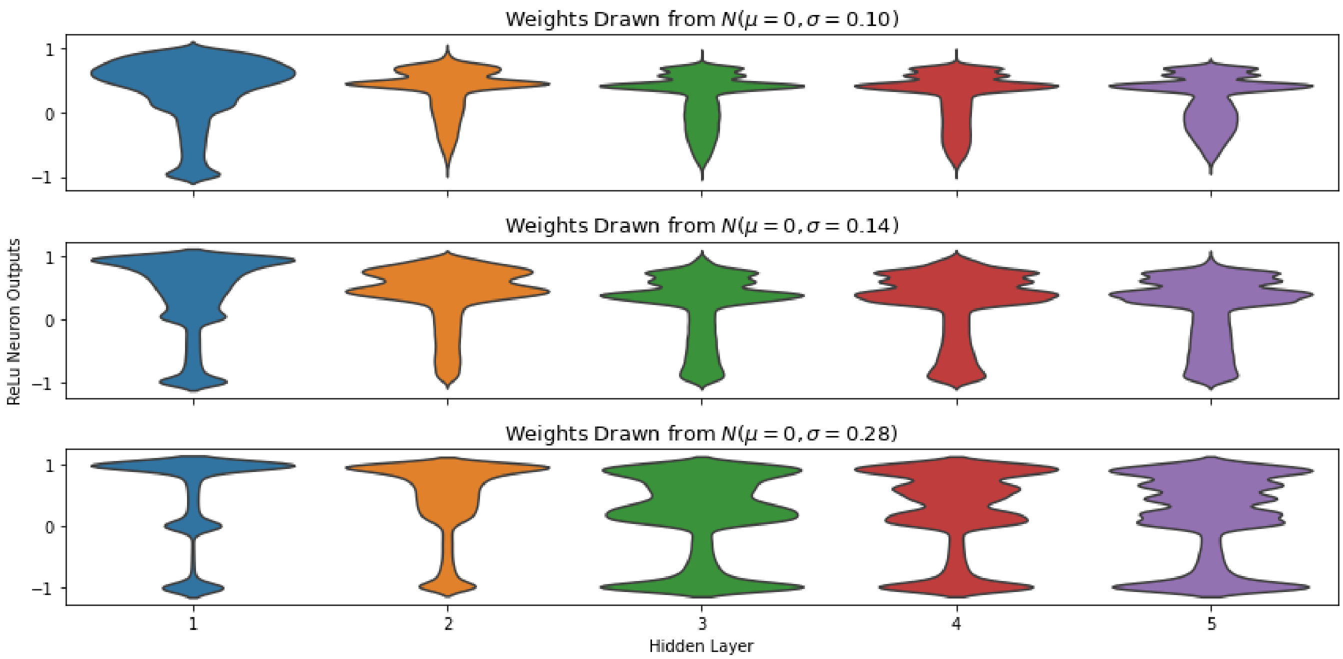
    ax.set_xlabel('')
    ax.set_ylabel('')

    ax.set_title('Weights Drawn from $N(\mu = 0, \sigma = {%.2f})$' % sig, fontsi
ze=13)

    if sig == sigmas[1]:
        ax.set_ylabel("ReLu Neuron Outputs")
    if sig != sigmas[-1]:
        ax.set_xticklabels(())
    else:
        ax.set_xlabel("Hidden Layer")

plt.tight_layout()
plt.show()
```

```
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
```



Use ReLU activation function with GlorotNormal initialization method

In [9]:
```
for stddev in sigmas:
    init = initializers.GlorotNormal(seed=seed)
    activation = 'relu'

    model = create_mlp_model(
        n_hidden_layers,
        dim_layer,
        (data_dim,),
        n_classes,
        init,
        'zeros',
        activation
    )
    compile_model(model)
    output_elts = get_activations(model, x_test)
    n_layers = len(model.layers)
    i_output_layer = n_layers - 1

    for i, out in enumerate(output_elts[:-1]):
        if i > 0 and i != i_output_layer:
            for out_i in out.ravel()[::20]:
                rows.append([i, stddev, out_i])

df = pd.DataFrame(rows, columns=['Hidden Layer', 'Standard Deviation', 'Output'])

# Plot previously saved activations from the 5 hidden layers
# using different initialization schemes.
fig = plt.figure(figsize=(12, 6))
axes = grid_axes_it(len(sigmas), 1, fig=fig)
for sig in sigmas:
    ax = next(axes)
    ddf = df[df['Standard Deviation'] == sig]
    sns.violinplot(x='Hidden Layer', y='Output', data=ddf, ax=ax, scale='count',
inner=None)

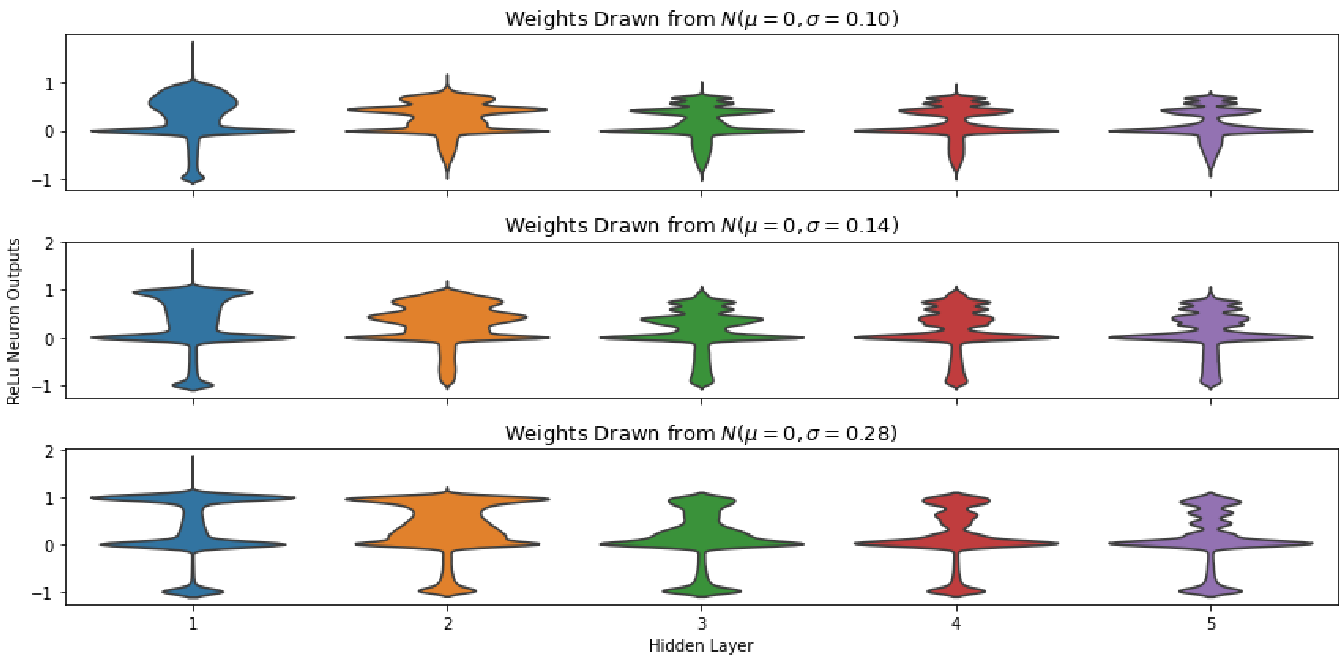
    ax.set_xlabel('')
    ax.set_ylabel('')

    ax.set_title('Weights Drawn from $N(\mu = 0, \sigma = {%.2f})$' % sig, fontsi
ze=13)

    if sig == sigmas[1]:
        ax.set_ylabel("ReLu Neuron Outputs")
    if sig != sigmas[-1]:
        ax.set_xticklabels(())
    else:
        ax.set_xlabel("Hidden Layer")

plt.tight_layout()
plt.show()
```

```
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
```



Use ReLU activation function with HeNormal initialization method

In [10]:
```
for stddev in sigmas:
    init = initializers.HeNormal(seed=None)
    activation = 'relu'

    model = create_mlp_model(
        n_hidden_layers,
        dim_layer,
        (data_dim,),
        n_classes,
        init,
        'zeros',
        activation
    )
    compile_model(model)
    output_elts = get_activations(model, x_test)
    n_layers = len(model.layers)
    i_output_layer = n_layers - 1

    for i, out in enumerate(output_elts[:-1]):
        if i > 0 and i != i_output_layer:
            for out_i in out.ravel()[::20]:
                rows.append([i, stddev, out_i])

df = pd.DataFrame(rows, columns=['Hidden Layer', 'Standard Deviation', 'Output'])

fig = plt.figure(figsize=(12, 6))
axes = grid_axes_it(len(sigmas), 1, fig=fig)
for sig in sigmas:
    ax = next(axes)
    ddf = df[df['Standard Deviation'] == sig]
    sns.violinplot(x='Hidden Layer', y='Output', data=ddf, ax=ax, scale='count',
inner=None)

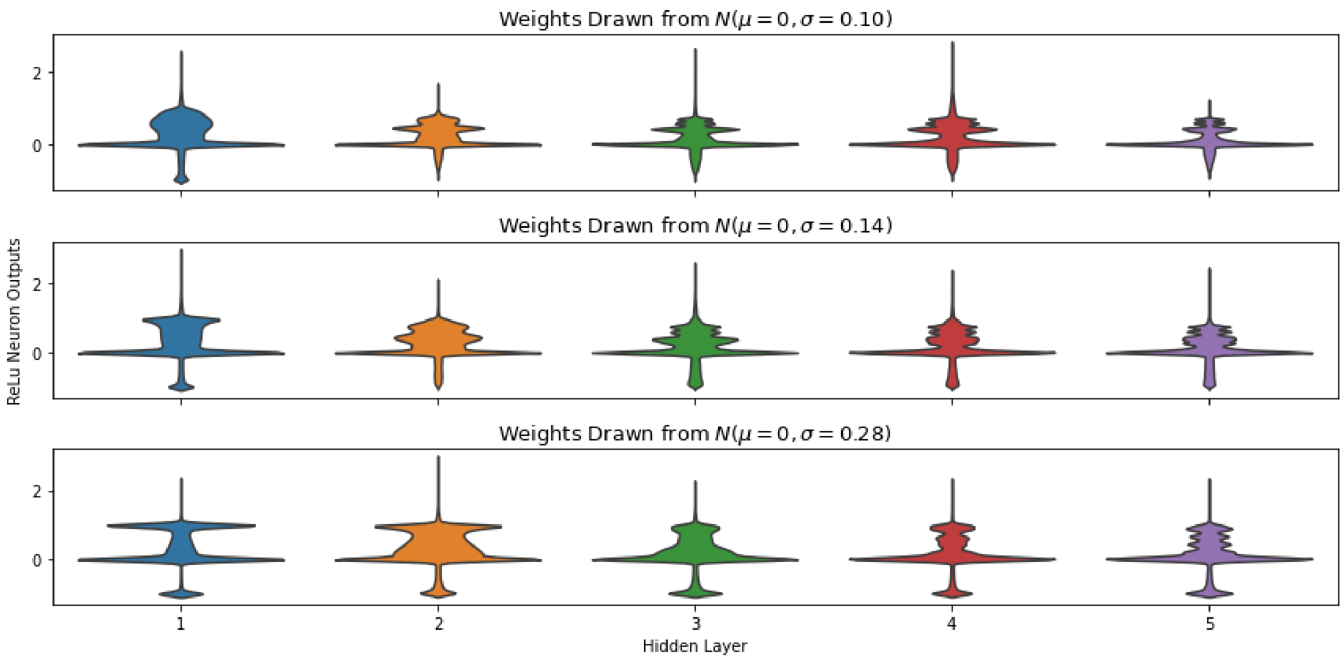
    ax.set_xlabel('')
    ax.set_ylabel('')

    ax.set_title('Weights Drawn from $N(\mu = 0, \sigma = {%.2f})$' % sig, fontsi
ze=13)

    if sig == sigmas[1]:
        ax.set_ylabel("ReLu Neuron Outputs")
    if sig != sigmas[-1]:
        ax.set_xticklabels(())
    else:
        ax.set_xlabel("Hidden Layer")

plt.tight_layout()
plt.show()
```

```
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
<ipython-input-6-a81fbcab33f4>:32: MatplotlibDeprecationWarning: Passing non-int
egers as three-element position specification is deprecated since 3.3 and will b
e removed two minor releases later.
  ax = plt.subplot(n_rows, n_cols, i)
```



Part 2

The dying ReLU is a kind of vanishing gradient, which refers to a problem when ReLU neurons become inactive and only output 0 for any input. In the worst case of dying ReLU, ReLU neurons at a certain layer are all dead, i.e., the entire network dies and is referred as the dying ReLU neural networks in Lu et al (reference below). A dying ReLU neural network collapses to a constant function. Show this phenomenon using any one of the three 1-dimensional functions in page 13 of Lu et al. Use a 10-layer ReLU network with width 2 (hidden units per layer). Use minibatch of 64 and draw training data uniformly from $[-\sqrt{7}, \sqrt{7}]$ Perform 1000 independent training simulations each with 3,000 training points. Out of these 1000 simulations, what fraction resulted in neural network collapse. Is your answer close to over 90% as was reported in Lu et al. ?

In [11]:
```
from keras import layers
from sklearn.model_selection import train_test_split
x = np.random.uniform(-np.sqrt(7),np.sqrt(7),3000)
y = x*np.sin(5*x)
x = x.reshape(-1,1)
y = y.reshape(-1,1)
init_method='HeNormal'
activation_method='relu'
# define the model
def create_model():
    model = keras.Sequential()
    model.add(layers.Dense(2, activation=activation_method,kernel_initializer=ini
t_method))
    model.add(layers.Dense(2, activation=activation_method,kernel_initializer=ini
t_method))
    model.add(layers.Dense(2, activation=activation_method,kernel_initializer=ini
t_method))
    model.add(layers.Dense(2, activation=activation_method,kernel_initializer=ini
t_method))
    model.add(layers.Dense(2, activation=activation_method,kernel_initializer=ini
t_method))
    model.add(layers.Dense(2, activation=activation_method,kernel_initializer=ini
t_method))
    model.add(layers.Dense(2, activation=activation_method,kernel_initializer=ini
t_method))
    model.add(layers.Dense(2, activation=activation_method,kernel_initializer=ini
t_method))
    model.add(layers.Dense(2, activation=activation_method,kernel_initializer=ini
t_method))
    model.add(layers.Dense(2, activation=activation_method,kernel_initializer=ini
t_method))
    model.add(layers.Dense(1,activation = activation_method))
    return model
```

In [12]:
```
preds=[]
for i in range(1000):
    if i%100==0:
        print(i)
    model=create_model()
    model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accurac
y'])
    model.fit(x=x,y=y, epochs=1, batch_size=64,verbose=0)
    pred=model.predict(x)[0]
    preds.append(pred)
```

```
0
100
200
300
400
500
600
700
800
900
```

In [13]:
```
# sum preds for each trail
preds_sum=[sum(i) for i in preds]
num_collapse =0
for i in preds_sum:
    if i==0.0:
        num_collapse+=1
print("number of neural network collapse:{}".format(num_collapse))
print("total number of trails:1000")
```

```
number of neural network collapse:1000
total number of trails:1000
```

Written Answer

- 100% (1000/1000) resulted in neural network collapse, which is over 90% as was reported in Lu et al.

Part 3

Instead of ReLU consider Leaky ReLU activation as defined below:

$$\phi(z) = \begin{cases} z & \text{if } z > 0 \\ 0.01z & \text{if } z \leq 0 \end{cases}$$

Run the 1000 training simulations in part 2 with Leaky ReLU activation and keeping everything else same. Again calculate the fraction of simulations that resulted in neural network collapse. Did Leaky ReLU help in preventing dying neurons ?

Code

In [18]:
```
import tensorflow as tf
def create_model_Leaky():
    model = keras.Sequential()
    model.add(layers.Dense(2))
    tf.keras.layers.LeakyReLU(alpha=0.3)
    model.add(layers.Dense(2))
    tf.keras.layers.LeakyReLU(alpha=0.3)
    model.add(layers.Dense(2))
    tf.keras.layers.LeakyReLU(alpha=0.3)
    model.add(layers.Dense(2))
    tf.keras.layers.LeakyReLU(alpha=0.3)
    model.add(layers.Dense(2))
    tf.keras.layers.LeakyReLU(alpha=0.3)
    model.add(layers.Dense(2))
    tf.keras.layers.LeakyReLU(alpha=0.3)
    model.add(layers.Dense(2))
    tf.keras.layers.LeakyReLU(alpha=0.3)
    model.add(layers.Dense(2))
    tf.keras.layers.LeakyReLU(alpha=0.3)
    model.add(layers.Dense(2))
    tf.keras.layers.LeakyReLU(alpha=0.3)
    model.add(layers.Dense(2))
    return model
```

```
In [19]: preds=[]
         for i in range(1000):
             if i%100==0:
                 print(i)
             model=create_model_Leaky()
             model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accurac
         y'])
             model.fit(x=x,y=y, epochs=1, batch_size=64,verbose=0)
             pred=model.predict(x)[0]
             preds.append(pred)
```

```
0
100
200
300
400
500
600
700
800
900
```

```
In [20]: # sum preds for each trail
         preds_sum=[sum(i) for i in preds]
         num_collapse =0
         for i in preds_sum:
             if i==0.0:
                 num_collapse+=1
         print("number of neural network collapse:{}".format(num_collapse))
         print("total number of trails:1000")
```

```
number of neural network collapse:0
total number of trails:1000
```

Written Answer

Only 0% (0/1000) resulted in neural network collapse. Leaky ReLU helped in preventing dying neurons.

Problem 2 - Batch Normalization, Dropout, MNIST

Part 1

Explain the terms co-adaptation and internal covariance-shift. Use examples if needed. You may need to refer to two papers mentioned below to answer this question. (4)

Written Answer

- Co-adaptation: It's when different hidden units in a neural networks have highly correlated behavior, which can be fixed by droupout
- Internal covariance-shift: occurs when there is a change in the input distribution to the nerual network.
  - When the input distribution changes, hidden layers try to learn to adapt to the new distribution, which slows down the training process.

Part 2

Batch normalization is traditionally used in hidden layers, for the input layer standard normalization is used. In standard normalization, the mean and standard deviation are calculated using the entire training dataset whereas in batch normalization these statistics are calculated for each mini-batch. Train LeNet-5 with standard normalization of input and batch normalization for hidden layers. What are the learned batch norm parameters for each layer? (4)

Code

download and preprocess mnist data

In [21]:
```
from keras.datasets import mnist
from keras.utils import np_utils
from tensorflow.keras.layers import BatchNormalization
import keras
import tensorflow as tf
# Load dataset as train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# change data type  and normalize
x_train = x_train.astype('float32')/255
x_test = x_test.astype('float32')/255

# Transform lables to one-hot encoding
y_train = keras.utils.np_utils.to_categorical(y_train, 10)
y_test = keras.utils.np_utils.to_categorical(y_test, 10)

# Reshape the dataset into 4D array
x_train = x_train.reshape(x_train.shape[0], 28,28,1)
x_test = x_test.reshape(x_test.shape[0], 28,28,1)
```

build model -> LeNet-5

In [22]:
```
from keras.models import Sequential
from keras import models, layers
import keras
#Instantiate an empty model
model = Sequential()
# C1 Convolutional Layer
model.add(layers.Conv2D(6, kernel_size=(5, 5), strides=(1, 1), activation='tanh',
input_shape=(28,28,1), padding='same'))
tf.keras.layers.LayerNormalization(trainable=True)
# S2 Pooling Layer
model.add(layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='vali
d'))
# C3 Convolutional Layer
model.add(layers.Conv2D(16, kernel_size=(5, 5), strides=(1, 1), activation='tanh'
, padding='valid'))
model.add(BatchNormalization(trainable=True))
# S4 Pooling Layer
model.add(layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='vali
d'))
# C5 Fully Connected Convolutional Layer
model.add(layers.Conv2D(120, kernel_size=(5, 5), strides=(1, 1), activation='tan
h', padding='valid'))
model.add(BatchNormalization(trainable=True))
#Flatten the CNN output so that we can connect it with fully connected layers
model.add(layers.Flatten())
# FC6 Fully Connected Layer
model.add(layers.Dense(84, activation='tanh'))
model.add(BatchNormalization(trainable=True))
#Output Layer with softmax activation
model.add(layers.Dense(10, activation='softmax'))
```

Train for 10 epochs

In [23]:
```
model.compile(loss=keras.losses.categorical_crossentropy, optimizer='SGD', metric
s=['accuracy'])
hist = model.fit(x=x_train,y=y_train, epochs=10, batch_size=128, validation_data=
(x_test, y_test), verbose=1)
test_score = model.evaluate(x_test, y_test)
print('Test loss {:.4f}, accuracy {:.2f}%'.format(test_score[0], test_score[1] *
100))
```

```
Epoch 1/10
469/469 [==============================] - 6s 9ms/step - loss: 0.3208 - accurac
y: 0.9097 - val_loss: 0.2330 - val_accuracy: 0.9407
Epoch 2/10
469/469 [==============================] - 4s 9ms/step - loss: 0.1697 - accurac
y: 0.9526 - val_loss: 0.1393 - val_accuracy: 0.9597
Epoch 3/10
469/469 [==============================] - 4s 9ms/step - loss: 0.1220 - accurac
y: 0.9656 - val_loss: 0.1008 - val_accuracy: 0.9707
Epoch 4/10
469/469 [==============================] - 4s 9ms/step - loss: 0.0988 - accurac
y: 0.9722 - val_loss: 0.0938 - val_accuracy: 0.9710
Epoch 5/10
469/469 [==============================] - 4s 9ms/step - loss: 0.0860 - accurac
y: 0.9758 - val_loss: 0.0784 - val_accuracy: 0.9768
Epoch 6/10
469/469 [==============================] - 4s 9ms/step - loss: 0.0762 - accurac
y: 0.9781 - val_loss: 0.0716 - val_accuracy: 0.9772
Epoch 7/10
469/469 [==============================] - 4s 9ms/step - loss: 0.0703 - accurac
y: 0.9800 - val_loss: 0.0648 - val_accuracy: 0.9802
Epoch 8/10
469/469 [==============================] - 4s 9ms/step - loss: 0.0650 - accurac
y: 0.9815 - val_loss: 0.0611 - val_accuracy: 0.9798
Epoch 9/10
469/469 [==============================] - 4s 9ms/step - loss: 0.0613 - accurac
y: 0.9831 - val_loss: 0.0751 - val_accuracy: 0.9755
Epoch 10/10
469/469 [==============================] - 4s 9ms/step - loss: 0.0567 - accurac
y: 0.9846 - val_loss: 0.0555 - val_accuracy: 0.9817
313/313 [==============================] - 1s 2ms/step - loss: 0.0555 - accurac
y: 0.9817
Test loss 0.0555, accuracy 98.17%
```

Get batch norm parameters for each layer

- Layer 4, 7, 10

In [24]:
```
for i in [3,6,9]:
    print("-------------")
    print(model.layers[i])
    print("Parameters", model.layers[i].get_weights()[0])
    print("bias",model.layers[i].get_weights()[1])
```

```
------------
<keras.layers.normalization.batch_normalization.BatchNormalization object at 0x1
5273e12f1c0>
Parameters [1.0262196 1.0343039 1.0227062 1.0456914 1.0336947 1.0336398 1.007473
2
 1.0726795 1.0460846 1.0642421 1.0712255 1.0196073 1.0153952 1.0230306
 1.0643497 1.0719744]
bias [ 0.00581085  0.00424867 -0.01398217 -0.0231855  -0.0021809   0.00325227
 -0.01482656  0.00916231 -0.02597473 -0.0038895  -0.02187813  0.00267884
 -0.00980926 -0.00209601 -0.01758475 -0.00101983]
------------
<keras.layers.normalization.batch_normalization.BatchNormalization object at 0x1
52362b945b0>
Parameters [1.0093138  0.99977946 0.9984202  1.0111787  0.9959657  1.0017868
 0.9929757  1.0012075  1.0038325  1.0063766  1.0167384  1.0098865
 0.9929446  1.0069867  0.99063045 1.0083458  1.0098206  1.0033276
 1.020129   1.0035548  1.0102153  1.0035807  1.0060563  1.003372
 1.0008101  1.0084469  1.0108656  1.0067018  1.00061    1.0017406
 0.99966353 1.003172   0.9984937  1.0053838  1.0048821  1.0002599
 0.9994941  0.9923873  1.0167812  1.0097134  0.9924526  1.00411
 1.0078907  1.0212317  1.0121369  1.000679   1.0197325  0.999018
 1.0053167  0.99878705 1.0027233  1.0020769  1.0058515  1.0178329
 1.0100886  0.99204576 1.0071948  1.0083625  0.9908136  0.99935615
 1.0101238  1.0046682  1.0028708  1.0141952  1.0015812  0.99760044
 0.99362373 1.0040267  1.0054958  1.0103892  0.995231   1.002626
 1.0021224  0.99923617 1.0028946  1.003717   0.99902797 1.007492
 1.0072298  1.0186347  1.0150889  1.0037453  0.99944097 1.0163407
 1.0242449  1.0026693  1.0017172  1.0026104  0.99926937 1.0183322
 0.9982721  1.0087622  1.0071119  1.0047125  0.9956982  1.0061184
 1.0069534  0.9963249  0.9991832  1.0125264  1.0073475  0.9964397
 1.0120759  0.98988485 1.0040133  1.007403   0.99652874 1.0048326
 1.0008409  1.0022266  1.0052669  1.0033638  0.99245656 1.0070174
 1.0031378  1.0117561  1.010205   1.0082963  1.0025623  1.007071  ]
bias [ 5.40142413e-03 -1.22030433e-02 -3.29574337e-03  1.02226855e-02
  1.33944973e-02 -2.25029435e-04 -8.84896424e-03  4.22716839e-03
 -4.99784015e-03 -3.54670477e-03  1.16743194e-02  7.87958317e-03
 -8.99233203e-03  5.90746803e-03 -4.38321754e-03  5.62806753e-03
 -1.56787317e-03  2.88397446e-03 -1.39245391e-02 -1.13660051e-02
  7.66305532e-03  4.32444183e-04 -5.25521813e-03 -1.18960785e-02
 -7.25596398e-03 -1.30104332e-03 -9.69763752e-03  1.01176994e-02
 -1.09460112e-03 -1.12909367e-02 -6.03059307e-03  7.29890168e-03
  7.08789937e-03 -2.64718011e-03 -1.95403676e-03 -6.99695153e-03
  1.19177857e-02  3.47729749e-03 -3.59613262e-03  6.88408967e-03
 -4.87571396e-03  1.05615275e-03 -2.68386235e-03  6.39555091e-03
 -4.31537628e-05 -8.66563246e-03 -2.41279093e-04 -1.26870509e-04
  4.51277010e-03 -5.16858045e-03  4.15794365e-03 -4.99999104e-03
  6.19095610e-03 -8.26130388e-04  6.73769566e-04 -8.62789620e-03
  1.17386719e-02  1.12336297e-02  7.23463250e-03 -6.03203243e-03
 -8.67852755e-03  2.80766212e-03 -9.14293714e-03  4.30731568e-03
  4.59127035e-03 -2.21986463e-03  5.82648441e-04 -4.30170010e-04
 -2.51853303e-03  5.31416014e-03 -4.33356687e-03 -2.09089555e-03
 -2.51743314e-03  6.37386367e-03  1.08282296e-02 -4.94494394e-04
  5.21167065e-04  8.54899548e-03  9.29094106e-03  3.22970259e-03
 -4.77413321e-03  8.44645407e-03  3.54542094e-03  6.45589409e-03
  2.79535330e-03 -1.48185901e-02  6.45353692e-03  7.29830656e-03
 -2.07624026e-03 -3.65637988e-03 -1.21015916e-02  9.16991755e-03
 -2.33100005e-03 -1.32000279e-02 -6.01610495e-03 -1.97343132e-03
 -3.78798228e-03  6.97111804e-03 -3.57916299e-03  7.85801373e-03
 -1.12254741e-02 -4.97817947e-03 -1.98884588e-03 -1.20721059e-03
```

```
 -5.87563962e-03 -6.64678577e-04 -2.25449400e-03 -4.23731515e-03
 -9.47097596e-03 -4.13644454e-03  5.19019272e-03 -1.07384345e-03
 -3.44498339e-03 -5.86610101e-03 -1.38018914e-02  4.77732625e-03
 -2.83911359e-03  3.48747033e-03  3.14582103e-05  2.39419891e-03]
------------
<keras.layers.normalization.batch_normalization.BatchNormalization object at 0x1
524d2ca6790>
Parameters [1.0245913 1.0238073 1.0182188 1.0337276 1.0495927 1.0078413 1.049095
5
 1.0648826 1.0384346 1.0307459 1.0711015 1.0496442 1.0245724 1.0328552
 1.025109  1.0299137 1.0436807 1.0548518 1.0583316 1.027606  1.0129274
 1.0447898 1.0436132 1.0369668 1.0154784 1.0556854 1.0277033 1.0352691
 1.0364426 1.0220166 1.0203578 1.0398933 1.0542699 1.0572743 1.0399398
 1.0377065 1.0219749 1.034119  1.0526437 1.0337161 1.0584646 1.0075783
 1.01656   1.0078996 1.0361679 1.0244309 1.0328317 1.050739  1.0456048
 1.0747086 1.0530667 1.0338757 1.0338069 1.0214258 1.0459088 1.0239484
 1.0466915 1.0561007 1.0242232 1.016687  1.013301  1.0370027 1.0455129
 1.0386589 1.0371622 1.047733  1.049823  1.0324498 1.0454259 1.0624388
 1.0434052 1.0211642 1.0541028 1.0445414 1.0418783 1.0235754 1.0220057
 1.022614  1.0548847 1.0425737 1.0280435 1.0236204 1.0411005 1.0398616]
bias [ 0.012996    0.01192081 -0.00855743  0.01664715 -0.00612286  0.00611061
 -0.01629852 -0.00753679  0.00881024  0.0043001   0.01007138 -0.00722215
  0.00891403  0.01765831 -0.00470379 -0.00296374 -0.0040032  -0.00586211
  0.00384224  0.00346095  0.00061971 -0.00336893 -0.00667123  0.00814271
  0.01656016  0.00016663 -0.00360594 -0.00478277 -0.01549278 -0.00483353
  0.00239336 -0.01988866  0.02223302  0.00873343 -0.00825373  0.00487204
  0.00554801  0.00400533 -0.01295513  0.00114712 -0.0058491  -0.00032001
 -0.01386327  0.0017028  -0.00468868 -0.00302274 -0.00148372  0.01651569
 -0.03071652  0.02209925  0.01003178  0.02073735  0.00029653  0.02404291
  0.00464291  0.0034426  -0.02230961 -0.02070673 -0.00894036  0.00395276
  0.00101628 -0.00693619  0.00879814 -0.00304667  0.00433259  0.01266176
  0.01482304  0.02134843  0.00905641 -0.01540268 -0.01524478 -0.0169195
  0.01974398  0.01676392  0.00925867 -0.00035231 -0.00855666 -0.00929621
  0.01567561  0.00766703 -0.00129983  0.01132998 -0.00804969  0.00182608]
```

Part 3

Next instead of standard normalization use batch normalization for the input layer also and train the network. Plot the distribution of learned batch norm parameters for each layer (including input) using violin plots. Compare the train/test accuracy and loss for the two cases? Did batch normalization for the input layer improve performance? (4)

Code

build model using batch normalization for the input layer

In [25]:
```
#Instantiate an empty model
model = Sequential()
model.add(layers.Conv2D(6, kernel_size=(5, 5), strides=(1, 1), activation='tanh', input_shape=(28,28,1), padding='same'))
model.add(BatchNormalization(trainable=True))
model.add(layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='valid'))
model.add(layers.Conv2D(16, kernel_size=(5, 5), strides=(1, 1), activation='tanh', padding='valid'))
model.add(BatchNormalization(trainable=True))
model.add(layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='valid'))
model.add(layers.Conv2D(120, kernel_size=(5, 5), strides=(1, 1), activation='tanh', padding='valid'))
model.add(BatchNormalization(trainable=True))
model.add(layers.Flatten())
model.add(layers.Dense(84, activation='tanh'))
model.add(BatchNormalization(trainable=True))
#Output Layer with softmax activation
model.add(layers.Dense(10, activation='softmax'))
```

Compile and train the model for 10 epochs

In [26]:
```
model.compile(loss=keras.losses.categorical_crossentropy, optimizer='SGD', metric
s=['accuracy'])

hist = model.fit(x=x_train,y=y_train, epochs=10, batch_size=128, validation_data=
(x_test, y_test), verbose=1)
test_score = model.evaluate(x_test, y_test)
print('Test loss {:.4f}, accuracy {:.2f}%'.format(test_score[0], test_score[1] *
100))
```

```
Epoch 1/10
469/469 [==============================] - 10s 21ms/step - loss: 0.3021 - accura
cy: 0.9155 - val_loss: 0.1976 - val_accuracy: 0.9469
Epoch 2/10
469/469 [==============================] - 10s 20ms/step - loss: 0.1431 - accura
cy: 0.9610 - val_loss: 0.1059 - val_accuracy: 0.9712
Epoch 3/10
469/469 [==============================] - 10s 21ms/step - loss: 0.1010 - accura
cy: 0.9718 - val_loss: 0.0843 - val_accuracy: 0.9763
Epoch 4/10
469/469 [==============================] - 10s 20ms/step - loss: 0.0802 - accura
cy: 0.9784 - val_loss: 0.0721 - val_accuracy: 0.9798
Epoch 5/10
469/469 [==============================] - 10s 20ms/step - loss: 0.0677 - accura
cy: 0.9812 - val_loss: 0.0667 - val_accuracy: 0.9813
Epoch 6/10
469/469 [==============================] - 10s 20ms/step - loss: 0.0590 - accura
cy: 0.9840 - val_loss: 0.0556 - val_accuracy: 0.9836
Epoch 7/10
469/469 [==============================] - 10s 20ms/step - loss: 0.0531 - accura
cy: 0.9850 - val_loss: 0.0544 - val_accuracy: 0.9839
Epoch 8/10
469/469 [==============================] - 10s 21ms/step - loss: 0.0481 - accura
cy: 0.9870 - val_loss: 0.0490 - val_accuracy: 0.9835
Epoch 9/10
469/469 [==============================] - 10s 21ms/step - loss: 0.0443 - accura
cy: 0.9880 - val_loss: 0.0449 - val_accuracy: 0.9856
Epoch 10/10
469/469 [==============================] - 10s 20ms/step - loss: 0.0420 - accura
cy: 0.9883 - val_loss: 0.0440 - val_accuracy: 0.9852
313/313 [==============================] - 1s 2ms/step - loss: 0.0440 - accurac
y: 0.9852
Test loss 0.0440, accuracy 98.52%
```

Plot the distribution of learned batch norm parameters for each layer

In [27]:
```
# 1, 4, 7, 10
model.layers
```

Out[27]:
```
[<keras.layers.convolutional.Conv2D at 0x152362b14eb0>,
 <keras.layers.normalization.batch_normalization.BatchNormalization at 0x1525e04
a0400>,
 <keras.layers.pooling.AveragePooling2D at 0x1523bc7ef610>,
 <keras.layers.convolutional.Conv2D at 0x1524d08ab340>,
 <keras.layers.normalization.batch_normalization.BatchNormalization at 0x1524d2c
a41f0>,
 <keras.layers.pooling.AveragePooling2D at 0x1523bc811a60>,
 <keras.layers.convolutional.Conv2D at 0x1524d08abbe0>,
 <keras.layers.normalization.batch_normalization.BatchNormalization at 0x1524d08
abf70>,
 <keras.layers.core.Flatten at 0x15258b687250>,
 <keras.layers.core.Dense at 0x1524d089cf70>,
 <keras.layers.normalization.batch_normalization.BatchNormalization at 0x1523bc8
29e50>,
 <keras.layers.core.Dense at 0x152363077eb0>]
```

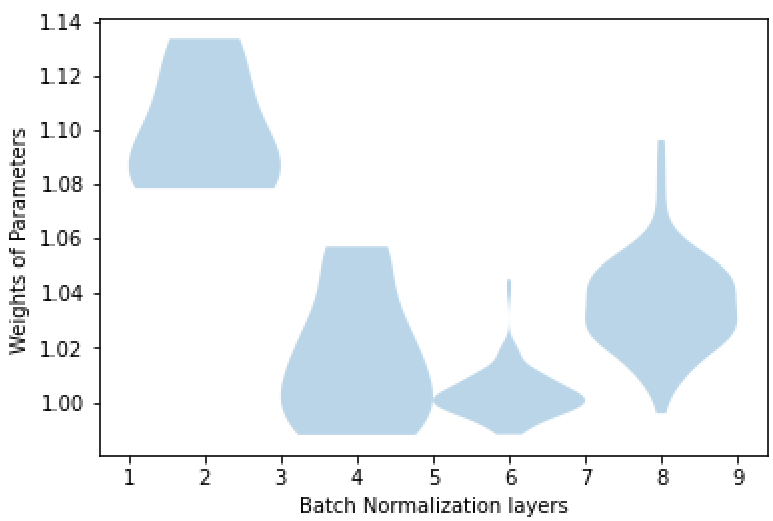
In [28]:
```
import matplotlib.pyplot as plt

# batch normalization layers
data=[]
for i in [1,4,7,10]:
    print(model.layers[i])
    data.append(model.layers[i].get_weights()[0])
fig, ax = plt.subplots()

ax.violinplot(data, [2, 4, 6,8],widths=2,
                  showmeans=False, showmedians=False, showextrema=False)
plt.xlabel("Batch Normalization layers")
plt.ylabel("Weights of Parameters ")
```

```
<keras.layers.normalization.batch_normalization.BatchNormalization object at 0x1
525e04a0400>
<keras.layers.normalization.batch_normalization.BatchNormalization object at 0x1
524d2ca41f0>
<keras.layers.normalization.batch_normalization.BatchNormalization object at 0x1
524d08abf70>
<keras.layers.normalization.batch_normalization.BatchNormalization object at 0x1
523bc829e50>
```

Out[28]:
```
Text(0, 0.5, 'Weights of Parameters ')
```



Written Answer

- Accuracy is improved, but not sigificantly.

Part 4

Train the network without batch normalization but this time use dropout. For hidden layers use a dropout probability of 0.5 and for input, layer take it to be 0.2 Compare test accuracy using dropout to test accuracy obtained using batch normalization in parts 2 and 3. (4)

Code

Build model with dropout

In [29]:
```
from keras.layers import Dropout
model = Sequential()
model.add(layers.Conv2D(6, kernel_size=(5, 5), strides=(1, 1), activation='tanh',
input_shape=(28,28,1), padding='same'))
model.add(Dropout(0.2))
model.add(layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='vali
d'))
model.add(Dropout(0.5))
model.add(layers.Conv2D(16, kernel_size=(5, 5), strides=(1, 1), activation='tanh'
, padding='valid'))
model.add(Dropout(0.5))
model.add(layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='vali
d'))
model.add(Dropout(0.5))
model.add(layers.Conv2D(120, kernel_size=(5, 5), strides=(1, 1), activation='tan
h', padding='valid'))
model.add(Dropout(0.5))
model.add(layers.Flatten())
model.add(layers.Dense(84, activation='tanh'))
model.add(Dropout(0.5))
model.add(layers.Dense(10, activation='softmax'))
```

Compile and train the model for 10 epochs

In [30]:
```
model.compile(loss=keras.losses.categorical_crossentropy, optimizer='SGD', metric
s=['accuracy'])
hist = model.fit(x=x_train,y=y_train, epochs=10, batch_size=128, validation_data=
(x_test, y_test), verbose=1)
test_score = model.evaluate(x_test, y_test)
print('Test loss {:.4f}, accuracy {:.2f}%'.format(test_score[0], test_score[1] *
100))
```

```
Epoch 1/10
469/469 [==============================] - 5s 10ms/step - loss: 1.7743 - accurac
y: 0.3967 - val_loss: 0.7976 - val_accuracy: 0.8249
Epoch 2/10
469/469 [==============================] - 5s 10ms/step - loss: 0.9919 - accurac
y: 0.6781 - val_loss: 0.4752 - val_accuracy: 0.8794
Epoch 3/10
469/469 [==============================] - 5s 10ms/step - loss: 0.8157 - accurac
y: 0.7334 - val_loss: 0.3975 - val_accuracy: 0.8916
Epoch 4/10
469/469 [==============================] - 5s 10ms/step - loss: 0.7420 - accurac
y: 0.7590 - val_loss: 0.3628 - val_accuracy: 0.8984
Epoch 5/10
469/469 [==============================] - 5s 10ms/step - loss: 0.7034 - accurac
y: 0.7741 - val_loss: 0.3425 - val_accuracy: 0.9032
Epoch 6/10
469/469 [==============================] - 5s 10ms/step - loss: 0.6779 - accurac
y: 0.7842 - val_loss: 0.3274 - val_accuracy: 0.9080
Epoch 7/10
469/469 [==============================] - 5s 10ms/step - loss: 0.6451 - accurac
y: 0.7955 - val_loss: 0.3159 - val_accuracy: 0.9106
Epoch 8/10
469/469 [==============================] - 5s 10ms/step - loss: 0.6288 - accurac
y: 0.8015 - val_loss: 0.3075 - val_accuracy: 0.9121
Epoch 9/10
469/469 [==============================] - 5s 10ms/step - loss: 0.6122 - accurac
y: 0.8076 - val_loss: 0.2978 - val_accuracy: 0.9151
Epoch 10/10
469/469 [==============================] - 5s 10ms/step - loss: 0.6007 - accurac
y: 0.8124 - val_loss: 0.2881 - val_accuracy: 0.9182
313/313 [==============================] - 1s 2ms/step - loss: 0.2881 - accurac
y: 0.9182
Test loss 0.2881, accuracy 91.82%
```

Written Answer

- Test accuracy using dropout decreased a lot from about 98% to 90%

Part 5

Now train the network using both batch normalization and dropout. How does the performance (test accuracy) of the network compare with the cases with dropout alone and with batch normalization alone? (4)

Code

Build the model using dropout and batch Normalization

In [31]:
```
model = Sequential()
model.add(layers.Conv2D(6, kernel_size=(5, 5), strides=(1, 1), activation='tanh',
input_shape=(28,28,1), padding='same'))
model.add(BatchNormalization(trainable=True))
model.add(Dropout(0.2))
model.add(layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='vali
d'))
model.add(Dropout(0.5))
model.add(layers.Conv2D(16, kernel_size=(5, 5), strides=(1, 1), activation='tanh'
, padding='valid'))
model.add(BatchNormalization(trainable=True))
model.add(Dropout(0.5))
model.add(layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='vali
d'))
model.add(Dropout(0.5))
model.add(layers.Conv2D(120, kernel_size=(5, 5), strides=(1, 1), activation='tan
h', padding='valid'))
model.add(BatchNormalization(trainable=True))
model.add(Dropout(0.5))
model.add(layers.Flatten())
model.add(layers.Dense(84, activation='tanh'))
model.add(BatchNormalization(trainable=True))
model.add(Dropout(0.5))
model.add(layers.Dense(10, activation='softmax'))
```

Compile the model and train for 10 epochs

In [32]:
```
model.compile(loss=keras.losses.categorical_crossentropy, optimizer='SGD', metric
s=['accuracy'])
hist = model.fit(x=x_train,y=y_train, epochs=10, batch_size=128, validation_data=
(x_test, y_test), verbose=1)
test_score = model.evaluate(x_test, y_test)
print('Test loss {:.4f}, accuracy {:.2f}%'.format(test_score[0], test_score[1] *
100))
```

```
Epoch 1/10
469/469 [==============================] - 11s 23ms/step - loss: 1.3322 - accura
cy: 0.5744 - val_loss: 0.3941 - val_accuracy: 0.8819
Epoch 2/10
469/469 [==============================] - 10s 22ms/step - loss: 0.8208 - accura
cy: 0.7352 - val_loss: 0.3324 - val_accuracy: 0.8990
Epoch 3/10
469/469 [==============================] - 10s 22ms/step - loss: 0.7123 - accura
cy: 0.7726 - val_loss: 0.3019 - val_accuracy: 0.9082
Epoch 4/10
469/469 [==============================] - 10s 22ms/step - loss: 0.6467 - accura
cy: 0.7968 - val_loss: 0.2779 - val_accuracy: 0.9138
Epoch 5/10
469/469 [==============================] - 10s 22ms/step - loss: 0.6030 - accura
cy: 0.8104 - val_loss: 0.2611 - val_accuracy: 0.9197
Epoch 6/10
469/469 [==============================] - 10s 22ms/step - loss: 0.5738 - accura
cy: 0.8183 - val_loss: 0.2432 - val_accuracy: 0.9253
Epoch 7/10
469/469 [==============================] - 10s 22ms/step - loss: 0.5432 - accura
cy: 0.8310 - val_loss: 0.2261 - val_accuracy: 0.9307
Epoch 8/10
469/469 [==============================] - 10s 22ms/step - loss: 0.5242 - accura
cy: 0.8353 - val_loss: 0.2158 - val_accuracy: 0.9324
Epoch 9/10
469/469 [==============================] - 10s 22ms/step - loss: 0.5015 - accura
cy: 0.8428 - val_loss: 0.2043 - val_accuracy: 0.9352
Epoch 10/10
469/469 [==============================] - 10s 22ms/step - loss: 0.4826 - accura
cy: 0.8484 - val_loss: 0.1932 - val_accuracy: 0.9397
313/313 [==============================] - 1s 2ms/step - loss: 0.1932 - accurac
y: 0.9397
Test loss 0.1932, accuracy 93.97%
```

Written Answer

- The accuracy is better than the performance with dropout, but worse than the performance with Batch Normalization

Problem 3 - Learning Rate, Batch Size, FashionMNIST

Helper functions from the keras and cyclic learning rate

In [5]:
```
import os

# initialize the list of class label names
CLASSES = ["top", "trouser", "pullover", "dress", "coat","sandal", "shirt", "snea
ker", "bag", "ankle boot"]

MIN_LR,MAX_LR = 1e-5,1e-2
BATCH_SIZE,STEP_SIZE = 64,8
```

In [6]:
```
from tensorflow.keras import backend as K
from tensorflow.keras.callbacks import *
import numpy as np

class CyclicLR(Callback):
        """This callback implements a cyclical learning rate policy (CLR).
        The method cycles the learning rate between two boundaries with
        some constant frequency, as detailed in this paper (https://arxiv.org/ab
s/1506.01186).
        The amplitude of the cycle can be scaled on a per-iteration or
        per-cycle basis.
        This class has three built-in policies, as put forth in the paper.
        "triangular":
                A basic triangular cycle w/ no amplitude scaling.
        "triangular2":
                A basic triangular cycle that scales initial amplitude by half ea
ch cycle.
        "exp_range":
                A cycle that scales initial amplitude by gamma**(cycle iteration
s) at each
                cycle iteration.
        For more detail, please see paper.

        # Example
                ```python
                        clr = CyclicLR(base_lr=0.001, max_lr=0.006,
                                                                step_size=2000.,
 mode='triangular')
                        model.fit(X_train, Y_train, callbacks=[clr])
                ```

        Class also supports custom scaling functions:
                ```python
                        clr_fn = lambda x: 0.5*(1+np.sin(x*np.pi/2.))
                        clr = CyclicLR(base_lr=0.001, max_lr=0.006,
                                                                step_size=2000.,
 scale_fn=clr_fn,
                                                                scale_mode='cycl
e')
                        model.fit(X_train, Y_train, callbacks=[clr])
                ```
        # Arguments
                base_lr: initial learning rate which is the
                        lower boundary in the cycle.
                max_lr: upper boundary in the cycle. Functionally,
                        it defines the cycle amplitude (max_lr - base_lr).
                        The lr at any cycle is the sum of base_lr
                        and some scaling of the amplitude; therefore
                        max_lr may not actually be reached depending on
                        scaling function.
                step_size: number of training iterations per
                        half cycle. Authors suggest setting step_size
                        2-8 x training iterations in epoch.
                mode: one of {triangular, triangular2, exp_range}.
                        Default 'triangular'.
                        Values correspond to policies detailed above.
                        If scale_fn is not None, this argument is ignored.
                gamma: constant in 'exp_range' scaling function:
                        gamma**(cycle iterations)
```

```
                scale_fn: Custom scaling policy defined by a single
                        argument lambda function, where
                        0 <= scale_fn(x) <= 1 for all x >= 0.
                        mode paramater is ignored
                scale_mode: {'cycle', 'iterations'}.
                        Defines whether scale_fn is evaluated on
                        cycle number or cycle iterations (training
                        iterations since start of cycle). Default is 'cycle'.
        """

        def __init__(self, base_lr=0.001, max_lr=0.006, step_size=2000., mode='triangular',
                                 gamma=1., scale_fn=None, scale_mode='cycle'):
                super(CyclicLR, self).__init__()

                self.base_lr = base_lr
                self.max_lr = max_lr
                self.step_size = step_size
                self.mode = mode
                self.gamma = gamma
                if scale_fn == None:
                        if self.mode == 'triangular':
                                self.scale_fn = lambda x: 1.
                                self.scale_mode = 'cycle'
                        elif self.mode == 'triangular2':
                                self.scale_fn = lambda x: 1 / (2. ** (x - 1))
                                self.scale_mode = 'cycle'
                        elif self.mode == 'exp_range':
                                self.scale_fn = lambda x: gamma ** (x)
                                self.scale_mode = 'iterations'
                else:
                        self.scale_fn = scale_fn
                        self.scale_mode = scale_mode
                self.clr_iterations = 0.
                self.trn_iterations = 0.
                self.history = {}

                self._reset()

        def _reset(self, new_base_lr=None, new_max_lr=None,
                           new_step_size=None):
                """Resets cycle iterations.
                Optional boundary/step size adjustment.
                """
                if new_base_lr != None:
                        self.base_lr = new_base_lr
                if new_max_lr != None:
                        self.max_lr = new_max_lr
                if new_step_size != None:
                        self.step_size = new_step_size
                self.clr_iterations = 0.

        def clr(self):
                cycle = np.floor(1 + self.clr_iterations / (2 * self.step_size))
                x = np.abs(self.clr_iterations / self.step_size - 2 * cycle + 1)
                if self.scale_mode == 'cycle':
                        return self.base_lr + (self.max_lr - self.base_lr) * np.maximum(0, (1 - x)) * self.scale_fn(cycle)
                else:
```

```
                        return self.base_lr + (self.max_lr - self.base_lr) * np.m
aximum(0, (1 - x)) * self.scale_fn(
                                self.clr_iterations)

        def on_train_begin(self, logs={}):
                logs = logs or {}

                if self.clr_iterations == 0:
                        K.set_value(self.model.optimizer.lr, self.base_lr)
                else:
                        K.set_value(self.model.optimizer.lr, self.clr())

        def on_batch_end(self, epoch, logs=None):

                logs = logs or {}
                self.trn_iterations += 1
                self.clr_iterations += 1

                self.history.setdefault('lr', []).append(K.get_value(self.model.o
ptimizer.lr))
                self.history.setdefault('iterations', []).append(self.trn_iterati
ons)

                for k, v in logs.items():
                        self.history.setdefault(k, []).append(v)

                K.set_value(self.model.optimizer.lr, self.clr())
```

In [7]:
```
# import the necessary packages
from tensorflow.keras.callbacks import LambdaCallback
from tensorflow.keras import backend as K
import matplotlib.pyplot as plt
import numpy as np
import tempfile

class LearningRateFinder:
        def __init__(self, model, stopFactor=4, beta=0.98):
                # store the model, stop factor, and beta value (for computing
                # a smoothed, average loss)
                self.model = model
                self.stopFactor = stopFactor
                self.beta = beta

                # initialize our list of learning rates and losses,
                # respectively
                self.lrs = []
                self.losses = []

                # initialize our learning rate multiplier, average loss, best
                # loss found thus far, current batch number, and weights file
                self.lrMult = 1
                self.avgLoss = 0
                self.bestLoss = 1e9
                self.batchNum = 0
                self.weightsFile = None

        def reset(self):
                # re-initialize all variables from our constructor
                self.lrs = []
                self.losses = []
                self.lrMult = 1
                self.avgLoss = 0
                self.bestLoss = 1e9
                self.batchNum = 0
                self.weightsFile = None

        def is_data_iter(self, data):
                # define the set of class types we will check for
                iterClasses = ["NumpyArrayIterator", "DirectoryIterator",
                         "Iterator", "Sequence"]

                # return whether our data is an iterator
                return data.__class__.__name__ in iterClasses

        def on_batch_end(self, batch, logs):
                # grab the current learning rate and add log it to the list of
                # learning rates that we've tried
                lr = K.get_value(self.model.optimizer.lr)
                self.lrs.append(lr)

                # grab the loss at the end of this batch, increment the total
                # number of batches processed, compute the average average
                # loss, smooth it, and update the losses list with the
                # smoothed value
                l = logs["loss"]
                self.batchNum += 1
                self.avgLoss = (self.beta * self.avgLoss) + ((1 - self.beta) * l)
```

```
            smooth = self.avgLoss / (1 - (self.beta ** self.batchNum))
            self.losses.append(smooth)

            # compute the maximum loss stopping factor value
            stopLoss = self.stopFactor * self.bestLoss

            # check to see whether the loss has grown too large
            if self.batchNum > 1 and smooth > stopLoss:
                    # stop returning and return from the method
                    self.model.stop_training = True
                    return

            # check to see if the best loss should be updated
            if self.batchNum == 1 or smooth < self.bestLoss:
                    self.bestLoss = smooth

            # increase the learning rate
            lr *= self.lrMult
            K.set_value(self.model.optimizer.lr, lr)

    def find(self, trainData, startLR, endLR, epochs=None,
            stepsPerEpoch=None, batchSize=32, sampleSize=2048,
            verbose=1):
            # reset our class-specific variables
            self.reset()

            # determine if we are using a data generator or not
            useGen = self.is_data_iter(trainData)

            # if we're using a generator and the steps per epoch is not
            # supplied, raise an error
            if useGen and stepsPerEpoch is None:
                    msg = "Using generator without supplying stepsPerEpoch"
                    raise Exception(msg)

            # if we're not using a generator then our entire dataset must
            # already be in memory
            elif not useGen:
                    # grab the number of samples in the training data and
                    # then derive the number of steps per epoch
                    numSamples = len(trainData[0])
                    stepsPerEpoch = np.ceil(numSamples / float(batchSize))

            # if no number of training epochs are supplied, compute the
            # training epochs based on a default sample size
            if epochs is None:
                    epochs = int(np.ceil(sampleSize / float(stepsPerEpoch)))

            # compute the total number of batch updates that will take
            # place while we are attempting to find a good starting
            # learning rate
            numBatchUpdates = epochs * stepsPerEpoch

            # derive the learning rate multiplier based on the ending
            # learning rate, starting learning rate, and total number of
            # batch updates
            self.lrMult = (endLR / startLR) ** (1.0 / numBatchUpdates)

            # create a temporary file path for the model weights and
```

```python
        # then save the weights (so we can reset the weights when we
        # are done)
        self.weightsFile = tempfile.mkstemp()[1]
        self.model.save_weights(self.weightsFile)

        # grab the *original* learning rate (so we can reset it
        # later), and then set the *starting* learning rate
        origLR = K.get_value(self.model.optimizer.lr)
        K.set_value(self.model.optimizer.lr, startLR)

        # construct a callback that will be called at the end of each
        # batch, enabling us to increase our learning rate as training
        # progresses
        callback = LambdaCallback(on_batch_end=lambda batch, logs:
                self.on_batch_end(batch, logs))

        # check to see if we are using a data iterator
        if useGen:
                self.model.fit(
                        x=trainData,
                        steps_per_epoch=stepsPerEpoch,
                        epochs=epochs,
                        verbose=verbose,
                        callbacks=[callback])

        # otherwise, our entire training data is already in memory
        else:
                # train our model using Keras' fit method
                self.model.fit(
                        x=trainData[0], y=trainData[1],
                        batch_size=batchSize,
                        epochs=epochs,
                        callbacks=[callback],
                        verbose=verbose)

        # restore the original model weights and learning rate
        self.model.load_weights(self.weightsFile)
        K.set_value(self.model.optimizer.lr, origLR)

def plot_loss(self, skipBegin=10, skipEnd=1, title=""):
        # grab the learning rate and losses values to plot
        lrs = self.lrs[skipBegin:-skipEnd]
        losses = self.losses[skipBegin:-skipEnd]

        # plot the learning rate vs. loss
        plt.plot(lrs, losses)
        plt.xscale("log")
        plt.xlabel("Learning Rate (Log Scale)")
        plt.ylabel("Loss")

        # if the title is not empty, add it to the plot
        if title != "":
                plt.title(title)
```

In [8]:
```
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Activation
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.layers import concatenate
from tensorflow.keras import backend as K

class MiniGoogLeNet:
        @staticmethod
        def conv_module(x, K, kX, kY, stride, chanDim, padding="same"):
                # define a CONV => BN => RELU pattern
                x = Conv2D(K, (kX, kY), strides=stride, padding=padding)(x)
                x = BatchNormalization(axis=chanDim)(x)
                x = Activation("relu")(x)

                # return the block
                return x

        @staticmethod
        def inception_module(x, numK1x1, numK3x3, chanDim):
                # define two CONV modules, then concatenate across the
                # channel dimension
                conv_1x1 = MiniGoogLeNet.conv_module(x, numK1x1, 1, 1,
                        (1, 1), chanDim)
                conv_3x3 = MiniGoogLeNet.conv_module(x, numK3x3, 3, 3,
                        (1, 1), chanDim)
                x = concatenate([conv_1x1, conv_3x3], axis=chanDim)

                # return the block
                return x

        @staticmethod
        def downsample_module(x, K, chanDim):
                # define the CONV module and POOL, then concatenate
                # across the channel dimensions
                conv_3x3 = MiniGoogLeNet.conv_module(x, K, 3, 3, (2, 2),
                        chanDim, padding="valid")
                pool = MaxPooling2D((3, 3), strides=(2, 2))(x)
                x = concatenate([conv_3x3, pool], axis=chanDim)

                # return the block
                return x

        @staticmethod
        def build(width, height, depth, classes):
                # initialize the input shape to be "channels last" and the
                # channels dimension itself
                inputShape = (height, width, depth)
                chanDim = -1

                # if we are using "channels first", update the input shape
                # and channels dimension
                if K.image_data_format() == "channels_first":
```

```
        inputShape = (depth, height, width)
        chanDim = 1

    # define the model input and first CONV module
    inputs = Input(shape=inputShape)
    x = MiniGoogLeNet.conv_module(inputs, 96, 3, 3, (1, 1),
        chanDim)

    # two Inception modules followed by a downsample module
    x = MiniGoogLeNet.inception_module(x, 32, 32, chanDim)
    x = MiniGoogLeNet.inception_module(x, 32, 48, chanDim)
    x = MiniGoogLeNet.downsample_module(x, 80, chanDim)

    # four Inception modules followed by a downsample module
    x = MiniGoogLeNet.inception_module(x, 112, 48, chanDim)
    x = MiniGoogLeNet.inception_module(x, 96, 64, chanDim)
    x = MiniGoogLeNet.inception_module(x, 80, 80, chanDim)
    x = MiniGoogLeNet.inception_module(x, 48, 96, chanDim)
    x = MiniGoogLeNet.downsample_module(x, 96, chanDim)

    # two Inception modules followed by global POOL and dropout
    x = MiniGoogLeNet.inception_module(x, 176, 160, chanDim)
    x = MiniGoogLeNet.inception_module(x, 176, 160, chanDim)
    x = AveragePooling2D((7, 7))(x)
    x = Dropout(0.5)(x)

    # softmax classifier
    x = Flatten()(x)
    x = Dense(classes)(x)
    x = Activation("softmax")(x)

    # create the model
    model = Model(inputs, x, name="googlenet")

    # return the constructed network architecture
    return model
```

Part 1

Fix batch size to 64 and start with 10 candidate learning rates between 10–9 and 101 and train your model for 5 epochs. Plot the training loss as a function of the learning rate. You should see a curve like Figure 3 in the reference below. From that figure identify the values of lrmin and lrmax. (2)

Code

In [10]:
```python
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.datasets import fashion_mnist
import matplotlib.pyplot as plt
import numpy as np
import argparse
import cv2
import sys
import tensorflow as tf
from tensorflow.compat.v1.keras.backend import set_session
config = tf.compat.v1.ConfigProto(device_count = {'GPU': 1 , 'CPU': 56} )
sess = tf.compat.v1.Session(config=config)
set_session(sess)
```

In [12]:
```python
print("load data")
((x_train, y_train), (x_test, y_test)) = fashion_mnist.load_data()

# resize to 32*32
x_train = np.array([cv2.resize(x, (32, 32)) for x in x_train])
x_test = np.array([cv2.resize(x, (32, 32)) for x in x_test])

# scale to the range [0, 1]
x_train = x_train.astype("float") / 255.0
x_test = x_test.astype("float") / 255.0


# reshape the data matrices to include a channel dimension
x_train = x_train.reshape((x_train.shape[0], 32, 32, 1))
x_test = x_test.reshape((x_test.shape[0], 32, 32, 1))


label_func = LabelBinarizer()
y_train = label_func.fit_transform(y_train)
y_test = label_func.transform(y_test)
# build image generator
data = ImageDataGenerator(width_shift_range=0.1, height_shift_range=0.1, horizont
al_flip=True, fill_mode="nearest")
```

```
load data
```

In [15]:
```python
# initialize the optimizer and model
print("compile model")
model = MiniGoogLeNet.build(width=32, height=32, depth=1, classes=10)
model.compile(loss="categorical_crossentropy", optimizer=SGD(lr=MIN_LR, momentum=
0.9),metrics=["accuracy"])


stepSize = STEP_SIZE * (x_train.shape[0] // BATCH_SIZE)
cyclic_lr = CyclicLR(
        mode='triangular',
        base_lr=1e-10,
        max_lr=10,
        step_size=10)

# train the network
print("training network")
history = model.fit(
        x=data.flow(x_train, y_train, batch_size=BATCH_SIZE),
        validation_data=(x_test, y_test),
        steps_per_epoch=x_train.shape[0] // BATCH_SIZE,
        epochs=5,
        callbacks=[cyclic_lr],
        verbose=1)
```

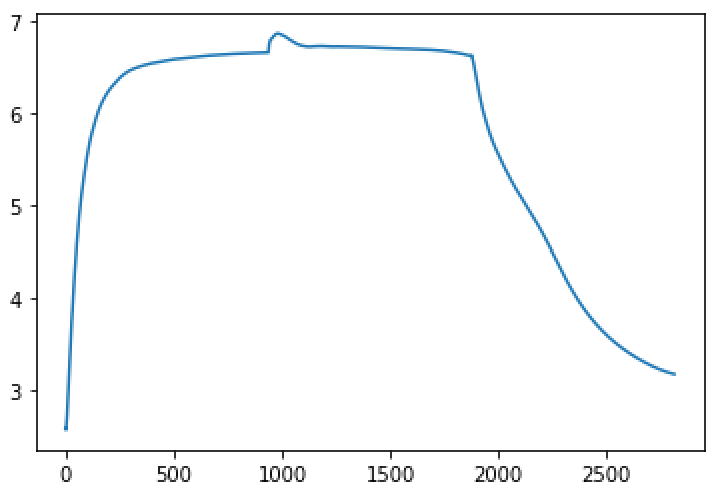
```
compile model

/home/zz3904/.local/lib/python3.8/site-packages/keras/optimizer_v2/optimizer_v2.
py:355: UserWarning: The `lr` argument is deprecated, use `learning_rate` instea
d.
  warnings.warn(

training network
Epoch 1/5
937/937 [==============================] - 351s 372ms/step - loss: 2.9404 - accu
racy: 0.1021 - val_loss: 3.0819 - val_accuracy: 0.1000
Epoch 2/5
937/937 [==============================] - 345s 369ms/step - loss: 2.7657 - accu
racy: 0.0993 - val_loss: 2.5275 - val_accuracy: 0.1000
Epoch 3/5
937/937 [==============================] - 346s 369ms/step - loss: 2.7222 - accu
racy: 0.0974 - val_loss: 2.4150 - val_accuracy: 0.1000
Epoch 4/5
937/937 [==============================] - 356s 380ms/step - loss: 2.7178 - accu
racy: 0.0995 - val_loss: 2.9067 - val_accuracy: 0.1000
Epoch 5/5
937/937 [==============================] - 363s 387ms/step - loss: 2.7061 - accu
racy: 0.1000 - val_loss: 2.5710 - val_accuracy: 0.1000
```

In [16]:
```
learning_rate_finder = LearningRateFinder(model)
learning_rate_finder.find(
                data.flow(x_train, y_train, batch_size=BATCH_SIZE),
                1e-10, 1e+1,
                stepsPerEpoch=np.ceil((len(x_train) / float(BATCH_SIZE))),
                batchSize=BATCH_SIZE)
```

```
Epoch 1/3
938/938 [==============================] - 351s 374ms/step - loss: 6.6644 - accu
racy: 0.1000
Epoch 2/3
938/938 [==============================] - 351s 375ms/step - loss: 6.5922 - accu
racy: 0.1000
Epoch 3/3
938/938 [==============================] - 355s 378ms/step - loss: 3.1382 - accu
racy: 0.0999
```

In [27]:
```
# plot the loss
from matplotlib import pyplot as plt
%matplotlib inline
plt.plot(learning_rate_finder.losses)
```

Out[27]: `[<matplotlib.lines.Line2D at 0x1516253fd610>]`



Part 2

Use the cyclical learning rate policy (with exponential decay) and train your network using batch size 64 and lrmin and lrmax values obtained in part 1. Plot train/validation loss and accuracy curve (similar to Figure 4 in reference). (3)

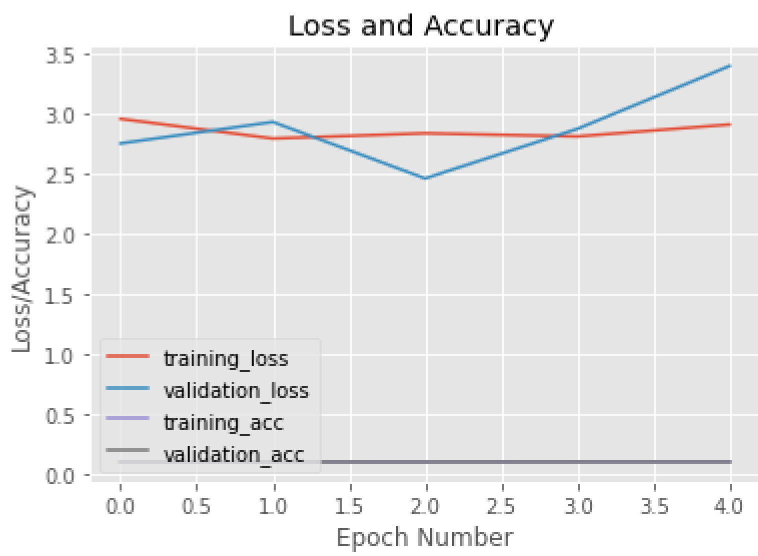
In [28]:

```python
lr_min = 1e-3
lr_max = 10

# initialize the optimizer and model
print("compile model...")
model = MiniGoogLeNet.build(width=32, height=32, depth=1, classes=10)
model.compile(loss="categorical_crossentropy", optimizer=SGD(learning_rate=lr_min
, momentum=0.9), metrics=["accuracy"])


stepSize = STEP_SIZE * (x_train.shape[0] // BATCH_SIZE)
cyclic_lr = CyclicLR(
        mode='triangular',
        base_lr=lr_min,
        max_lr=lr_max,
        step_size=STEP_SIZE)


# train the network
print("train the model")
history = model.fit(
        x=data.flow(x_train, y_train, batch_size=BATCH_SIZE),
        validation_data=(x_test, y_test),
        steps_per_epoch=x_train.shape[0] // BATCH_SIZE,
        epochs=5,
        callbacks=[cyclic_lr],
        verbose=1)
```

```
compile model...
train the model
Epoch 1/5
937/937 [==============================] - 351s 373ms/step - loss: 2.9595 - accu
racy: 0.1004 - val_loss: 2.7541 - val_accuracy: 0.1000
Epoch 2/5
937/937 [==============================] - 346s 370ms/step - loss: 2.7952 - accu
racy: 0.0988 - val_loss: 2.9332 - val_accuracy: 0.1000
Epoch 3/5
937/937 [==============================] - 347s 370ms/step - loss: 2.8389 - accu
racy: 0.0984 - val_loss: 2.4615 - val_accuracy: 0.1000
Epoch 4/5
937/937 [==============================] - 343s 366ms/step - loss: 2.8126 - accu
racy: 0.1007 - val_loss: 2.8757 - val_accuracy: 0.1000
Epoch 5/5
937/937 [==============================] - 268s 286ms/step - loss: 2.9120 - accu
racy: 0.1007 - val_loss: 3.4010 - val_accuracy: 0.1000
```

In [29]:
```
# Plot train/validation loss and accuracy curve
plt_loc="lower left"
size = np.arange(0, 5)
plt.style.use("ggplot")
plt.figure()
plt.plot(size, history.history["loss"], label="training_loss")
plt.plot(size, history.history["val_loss"], label="validation_loss")
plt.plot(size, history.history["accuracy"], label="training_acc")
plt.plot(size, history.history["val_accuracy"], label="validation_acc")
plt.title("Loss and Accuracy ")
plt.xlabel("Epoch Number")
plt.ylabel("Loss/Accuracy")
plt.legend(loc=plt_loc)
```

Out[29]:
```
<matplotlib.legend.Legend at 0x15161e809160>
```



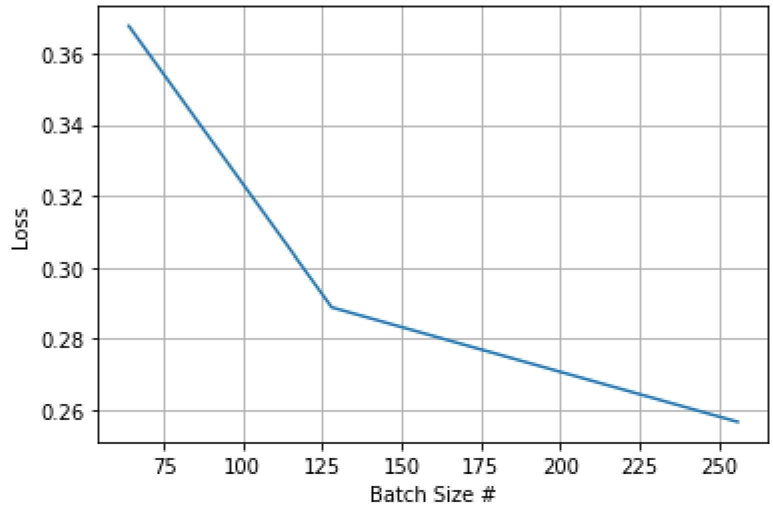
Part 3

We want to test if increasing batch size for a fixed learning rate has the same effect as decreasing learning rate for fixed batch size. Fix learning rate to lrmax and train your network starting with batch size 32 and incrementally going up to 16384 (in increments of a factor of 2; like 32, 64...). You can choose a step size (in terms of the number of iterations) to increment the batch size. If your GPU cannot handle large batch sizes, you need to employ an effective batch size approach as discussed in Lecture 3 to simulate large batches. Plot the training loss. Is the generalization of your final model similar or different from than cyclical learning rate policy? (10)

In [13]:
```
lr_min = 1e-3
lr_max = 1e-3
step_size = 1

# initialize the optimizer and model
print("compile model")
model = MiniGoogLeNet.build(width=32, height=32, depth=1, classes=10)
model.compile(loss="categorical_crossentropy", optimizer=SGD(learning_rate=lr_min
, momentum=0.9),metrics=["accuracy"])

batch_size_list = [64,128,256]
loss_arr = []

for i in batch_size_list:
    stepSize = step_size * (x_train.shape[0] // i)
    cyclic_lr = CyclicLR(
        mode='triangular',
        base_lr=lr_min,
        max_lr=lr_max,
        step_size=step_size)

    # train the model
    print("training","batch size: {}".format(i))
    history = model.fit(
        x=data.flow(x_train, y_train, batch_size=i),
        validation_data=(x_test, y_test),
        steps_per_epoch=x_train.shape[0] // i,
        epochs=5,
        callbacks=[cyclic_lr],
        verbose=1)
    loss_arr.append(history.history["loss"][4])
```

```
compile model
training batch size: 64
Epoch 1/5
937/937 [==============================] - 241s 254ms/step - loss: 0.8526 - accu
racy: 0.6966 - val_loss: 0.6672 - val_accuracy: 0.7576
Epoch 2/5
937/937 [==============================] - 333s 355ms/step - loss: 0.5220 - accu
racy: 0.8094 - val_loss: 0.4759 - val_accuracy: 0.8272
Epoch 3/5
937/937 [==============================] - 361s 385ms/step - loss: 0.4490 - accu
racy: 0.8388 - val_loss: 0.4280 - val_accuracy: 0.8461
Epoch 4/5
937/937 [==============================] - 358s 382ms/step - loss: 0.4007 - accu
racy: 0.8546 - val_loss: 0.4408 - val_accuracy: 0.8419
Epoch 5/5
937/937 [==============================] - 364s 389ms/step - loss: 0.3678 - accu
racy: 0.8676 - val_loss: 0.3707 - val_accuracy: 0.8678
training batch size: 128
Epoch 1/5
468/468 [==============================] - 362s 773ms/step - loss: 0.3268 - accu
racy: 0.8808 - val_loss: 0.3144 - val_accuracy: 0.8903
Epoch 2/5
468/468 [==============================] - 360s 769ms/step - loss: 0.3113 - accu
racy: 0.8878 - val_loss: 0.3392 - val_accuracy: 0.8785
Epoch 3/5
468/468 [==============================] - 364s 777ms/step - loss: 0.3025 - accu
racy: 0.8914 - val_loss: 0.3224 - val_accuracy: 0.8836
Epoch 4/5
468/468 [==============================] - 371s 792ms/step - loss: 0.2949 - accu
racy: 0.8945 - val_loss: 0.3561 - val_accuracy: 0.8730
Epoch 5/5
468/468 [==============================] - 372s 794ms/step - loss: 0.2888 - accu
racy: 0.8962 - val_loss: 0.2974 - val_accuracy: 0.8898
training batch size: 256
Epoch 1/5
234/234 [==============================] - 377s 2s/step - loss: 0.2695 - accurac
y: 0.9053 - val_loss: 0.2731 - val_accuracy: 0.8999
Epoch 2/5
234/234 [==============================] - 377s 2s/step - loss: 0.2645 - accurac
y: 0.9048 - val_loss: 0.3151 - val_accuracy: 0.8883
Epoch 3/5
234/234 [==============================] - 368s 2s/step - loss: 0.2624 - accurac
y: 0.9066 - val_loss: 0.2723 - val_accuracy: 0.9023
Epoch 4/5
234/234 [==============================] - 361s 2s/step - loss: 0.2577 - accurac
y: 0.9081 - val_loss: 0.2702 - val_accuracy: 0.9026
Epoch 5/5
234/234 [==============================] - 362s 2s/step - loss: 0.2566 - accurac
y: 0.9080 - val_loss: 0.3091 - val_accuracy: 0.8898
```

In [14]:
```
from matplotlib import pyplot as plt
plt.plot(batch_size_list, np.asarray(loss_arr))
plt.xlabel('Batch Size #')
plt.ylabel('Loss')
plt.grid()
```



Problem 4 - Adaptive Learning Rate Methods, CIFAR-10

Part 1

Write the weight update equations for the five adaptive learning rate methods. Explain each term clearly. What are the hyperparameters in each policy? Explain how AdaDelta and Adam are different from RMSProp. (5+1)

Written Answer

AdaGrad

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

- θ: the parameter
- η: the learning rate
- ϵ: a smoothing term that prevents division by 0
- $\odot$: the matrix-vector product
- g_t: the gradient at time t step
- G_t: a diagonal matrix where each diagonal element i, i is the sum of squares of the gradients with respect to θ up to time step t.

RMSProp

$$\begin{aligned} E[g^2]_t &= 0.9E[g^2]_{t-1} + 0.1g_t^2 \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \end{aligned}$$

- $E[g^2]_t$: the moving average of squared gradients
- g_t^2: gradient of the cost function with respect to the weight
- η: the learning rate
- θ_{t+1}: the parameter
- 0.9: moving average parameter

RMSProp+Nesterov

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}(\beta_1 \hat{m}_t + \frac{(1-\beta_1)g_t}{1-\beta_1^t})$$

- γ is the momentum decay term. β_1 is the decay rate.

AdaDelta

$$\begin{aligned} \Delta\theta_t &= -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t \\ \theta_{t+1} &= \theta_t + \Delta\theta_t \end{aligned}$$

- RMS: the root mean squaured error
- g_t: gradient of the cost function with respect to the weight

RMS

- the root mean squared error

Adam

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

- m_t estimates of the first moment vector of the gradients
- v_t: estimates ofthe second moment of the gradients
- ϵ: a small positive constant

Explain how AdaDelta and Adam are different from RMSProp.

- RMSprop is an extension of Adagrad that deals with its radically diminishing learning rates.
- RMSprop is basically similiar to Adadelta. The only difference is Adadelta uess the RMS of parameter updates in the numinator to update rule.
- Adam is different from RMSProp in the way that adds bias-correction and momentum to RMSprop.

Part 2

Train the neural network using all the five methods with L2-regularization for 200 epochs each and plot the training loss vs the number of epochs. Which method performs best (lowest training loss)? (5)

Code

Load Data

In [15]:
```
from keras.datasets import cifar10
import tensorflow as tf
from tensorflow.compat.v1.keras.backend import set_session
config = tf.compat.v1.ConfigProto(device_count = {'GPU': 1 , 'CPU': 56} )
sess = tf.compat.v1.Session(config=config)
set_session(sess)


#import dataset
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

#Preprocessing
y_train = tf.keras.utils.to_categorical(y_train)
y_test = tf.keras.utils.to_categorical(y_test)
X_train = X_train.astype('float32')/255.0
X_test = X_test.astype('float32')/255.0
```

Build model witout dropout

In [16]:
```
from keras import Sequential
from keras.layers import Flatten, Dense
activation_method='relu'
kregularizer_method='l2'
init_method = 'HeNormal'
def create_model():
    model = Sequential()
    model.add(Flatten())
    model.add(Dense(1000,activation=activation_method,kernel_regularizer=kregular
izer_method,kernel_initializer=init_method))
    model.add(Dense(1000,activation=activation_method,kernel_regularizer=kregular
izer_method,kernel_initializer=init_method))
    model.add(Dense(10, activation='softmax'))
    return model
```

Adagrad

In [17]:

```
model_adagrad=create_model()
model_adagrad.compile(loss='categorical_crossentropy', optimizer='Adagrad', metri
cs=['accuracy'])
history_adagrad = model_adagrad.fit(X_train, y_train, batch_size=128, epochs=200,
validation_data=(X_test,y_test),shuffle=True,verbose=2)
```

```
Epoch 1/200
391/391 - 6s - loss: 41.0613 - accuracy: 0.2899 - val_loss: 39.9960 - val_accura
cy: 0.3446
Epoch 2/200
391/391 - 5s - loss: 39.0420 - accuracy: 0.3543 - val_loss: 38.1103 - val_accura
cy: 0.3653
Epoch 3/200
391/391 - 5s - loss: 37.2241 - accuracy: 0.3739 - val_loss: 36.3511 - val_accura
cy: 0.3853
Epoch 4/200
391/391 - 5s - loss: 35.5205 - accuracy: 0.3868 - val_loss: 34.7003 - val_accura
cy: 0.3901
Epoch 5/200
391/391 - 5s - loss: 33.9142 - accuracy: 0.3950 - val_loss: 33.1414 - val_accura
cy: 0.4019
Epoch 6/200
391/391 - 5s - loss: 32.3959 - accuracy: 0.4031 - val_loss: 31.6655 - val_accura
cy: 0.4070
Epoch 7/200
391/391 - 5s - loss: 30.9590 - accuracy: 0.4091 - val_loss: 30.2701 - val_accura
cy: 0.4153
Epoch 8/200
391/391 - 5s - loss: 29.5976 - accuracy: 0.4145 - val_loss: 28.9455 - val_accura
cy: 0.4153
Epoch 9/200
391/391 - 5s - loss: 28.3064 - accuracy: 0.4187 - val_loss: 27.6849 - val_accura
cy: 0.4182
Epoch 10/200
391/391 - 5s - loss: 27.0817 - accuracy: 0.4224 - val_loss: 26.5010 - val_accura
cy: 0.4157
Epoch 11/200
391/391 - 5s - loss: 25.9194 - accuracy: 0.4259 - val_loss: 25.3655 - val_accura
cy: 0.4216
Epoch 12/200
391/391 - 5s - loss: 24.8161 - accuracy: 0.4271 - val_loss: 24.2918 - val_accura
cy: 0.4226
Epoch 13/200
391/391 - 5s - loss: 23.7672 - accuracy: 0.4299 - val_loss: 23.2728 - val_accura
cy: 0.4228
Epoch 14/200
391/391 - 5s - loss: 22.7710 - accuracy: 0.4323 - val_loss: 22.2949 - val_accura
cy: 0.4299
Epoch 15/200
391/391 - 5s - loss: 21.8232 - accuracy: 0.4359 - val_loss: 21.3729 - val_accura
cy: 0.4269
Epoch 16/200
391/391 - 5s - loss: 20.9215 - accuracy: 0.4393 - val_loss: 20.4974 - val_accura
cy: 0.4310
Epoch 17/200
391/391 - 5s - loss: 20.0647 - accuracy: 0.4404 - val_loss: 19.6632 - val_accura
cy: 0.4298
Epoch 18/200
391/391 - 5s - loss: 19.2492 - accuracy: 0.4427 - val_loss: 18.8623 - val_accura
cy: 0.4380
Epoch 19/200
391/391 - 5s - loss: 18.4725 - accuracy: 0.4431 - val_loss: 18.1084 - val_accura
cy: 0.4414
Epoch 20/200
391/391 - 5s - loss: 17.7328 - accuracy: 0.4457 - val_loss: 17.3849 - val_accura
```

```
cy: 0.4405
Epoch 21/200
391/391 - 5s - loss: 17.0290 - accuracy: 0.4464 - val_loss: 16.6982 - val_accura
cy: 0.4425
Epoch 22/200
391/391 - 5s - loss: 16.3584 - accuracy: 0.4467 - val_loss: 16.0489 - val_accura
cy: 0.4374
Epoch 23/200
391/391 - 5s - loss: 15.7194 - accuracy: 0.4499 - val_loss: 15.4242 - val_accura
cy: 0.4436
Epoch 24/200
391/391 - 5s - loss: 15.1100 - accuracy: 0.4503 - val_loss: 14.8291 - val_accura
cy: 0.4411
Epoch 25/200
391/391 - 5s - loss: 14.5293 - accuracy: 0.4524 - val_loss: 14.2609 - val_accura
cy: 0.4481
Epoch 26/200
391/391 - 5s - loss: 13.9758 - accuracy: 0.4528 - val_loss: 13.7201 - val_accura
cy: 0.4451
Epoch 27/200
391/391 - 5s - loss: 13.4481 - accuracy: 0.4548 - val_loss: 13.2035 - val_accura
cy: 0.4436
Epoch 28/200
391/391 - 5s - loss: 12.9445 - accuracy: 0.4541 - val_loss: 12.7136 - val_accura
cy: 0.4448
Epoch 29/200
391/391 - 5s - loss: 12.4641 - accuracy: 0.4562 - val_loss: 12.2462 - val_accura
cy: 0.4461
Epoch 30/200
391/391 - 5s - loss: 12.0060 - accuracy: 0.4580 - val_loss: 11.8020 - val_accura
cy: 0.4506
Epoch 31/200
391/391 - 5s - loss: 11.5686 - accuracy: 0.4579 - val_loss: 11.3727 - val_accura
cy: 0.4439
Epoch 32/200
391/391 - 5s - loss: 11.1507 - accuracy: 0.4588 - val_loss: 10.9632 - val_accura
cy: 0.4518
Epoch 33/200
391/391 - 5s - loss: 10.7524 - accuracy: 0.4594 - val_loss: 10.5754 - val_accura
cy: 0.4500
Epoch 34/200
391/391 - 5s - loss: 10.3718 - accuracy: 0.4612 - val_loss: 10.2030 - val_accura
cy: 0.4517
Epoch 35/200
391/391 - 5s - loss: 10.0087 - accuracy: 0.4604 - val_loss: 9.8491 - val_accurac
y: 0.4484
Epoch 36/200
391/391 - 5s - loss: 9.6613 - accuracy: 0.4625 - val_loss: 9.5073 - val_accurac
y: 0.4529
Epoch 37/200
391/391 - 5s - loss: 9.3296 - accuracy: 0.4612 - val_loss: 9.1878 - val_accurac
y: 0.4536
Epoch 38/200
391/391 - 5s - loss: 9.0129 - accuracy: 0.4638 - val_loss: 8.8747 - val_accurac
y: 0.4536
Epoch 39/200
391/391 - 5s - loss: 8.7097 - accuracy: 0.4640 - val_loss: 8.5845 - val_accurac
y: 0.4522
Epoch 40/200
```

```
391/391 - 5s - loss: 8.4206 - accuracy: 0.4641 - val_loss: 8.3004 - val_accurac
y: 0.4540
Epoch 41/200
391/391 - 5s - loss: 8.1440 - accuracy: 0.4656 - val_loss: 8.0286 - val_accurac
y: 0.4546
Epoch 42/200
391/391 - 5s - loss: 7.8803 - accuracy: 0.4658 - val_loss: 7.7663 - val_accurac
y: 0.4570
Epoch 43/200
391/391 - 5s - loss: 7.6272 - accuracy: 0.4670 - val_loss: 7.5192 - val_accurac
y: 0.4605
Epoch 44/200
391/391 - 5s - loss: 7.3846 - accuracy: 0.4676 - val_loss: 7.2894 - val_accurac
y: 0.4579
Epoch 45/200
391/391 - 5s - loss: 7.1536 - accuracy: 0.4683 - val_loss: 7.0558 - val_accurac
y: 0.4613
Epoch 46/200
391/391 - 5s - loss: 6.9325 - accuracy: 0.4684 - val_loss: 6.8426 - val_accurac
y: 0.4570
Epoch 47/200
391/391 - 5s - loss: 6.7212 - accuracy: 0.4679 - val_loss: 6.6335 - val_accurac
y: 0.4617
Epoch 48/200
391/391 - 5s - loss: 6.5182 - accuracy: 0.4700 - val_loss: 6.4361 - val_accurac
y: 0.4611
Epoch 49/200
391/391 - 5s - loss: 6.3245 - accuracy: 0.4693 - val_loss: 6.2466 - val_accurac
y: 0.4606
Epoch 50/200
391/391 - 5s - loss: 6.1391 - accuracy: 0.4703 - val_loss: 6.0708 - val_accurac
y: 0.4598
Epoch 51/200
391/391 - 5s - loss: 5.9613 - accuracy: 0.4707 - val_loss: 5.8952 - val_accurac
y: 0.4612
Epoch 52/200
391/391 - 5s - loss: 5.7915 - accuracy: 0.4719 - val_loss: 5.7265 - val_accurac
y: 0.4643
Epoch 53/200
391/391 - 5s - loss: 5.6288 - accuracy: 0.4719 - val_loss: 5.5682 - val_accurac
y: 0.4652
Epoch 54/200
391/391 - 5s - loss: 5.4733 - accuracy: 0.4737 - val_loss: 5.4174 - val_accurac
y: 0.4552
Epoch 55/200
391/391 - 5s - loss: 5.3236 - accuracy: 0.4719 - val_loss: 5.2700 - val_accurac
y: 0.4642
Epoch 56/200
391/391 - 5s - loss: 5.1808 - accuracy: 0.4737 - val_loss: 5.1297 - val_accurac
y: 0.4638
Epoch 57/200
391/391 - 5s - loss: 5.0437 - accuracy: 0.4744 - val_loss: 4.9993 - val_accurac
y: 0.4608
Epoch 58/200
391/391 - 5s - loss: 4.9125 - accuracy: 0.4758 - val_loss: 4.8671 - val_accurac
y: 0.4687
Epoch 59/200
391/391 - 5s - loss: 4.7865 - accuracy: 0.4757 - val_loss: 4.7482 - val_accurac
y: 0.4624
```

```
Epoch 60/200
391/391 - 5s - loss: 4.6664 - accuracy: 0.4764 - val_loss: 4.6259 - val_accurac
y: 0.4669
Epoch 61/200
391/391 - 5s - loss: 4.5507 - accuracy: 0.4760 - val_loss: 4.5168 - val_accurac
y: 0.4671
Epoch 62/200
391/391 - 5s - loss: 4.4401 - accuracy: 0.4770 - val_loss: 4.4047 - val_accurac
y: 0.4701
Epoch 63/200
391/391 - 5s - loss: 4.3340 - accuracy: 0.4786 - val_loss: 4.3044 - val_accurac
y: 0.4617
Epoch 64/200
391/391 - 5s - loss: 4.2320 - accuracy: 0.4792 - val_loss: 4.2009 - val_accurac
y: 0.4672
Epoch 65/200
391/391 - 5s - loss: 4.1346 - accuracy: 0.4795 - val_loss: 4.1057 - val_accurac
y: 0.4689
Epoch 66/200
391/391 - 5s - loss: 4.0408 - accuracy: 0.4795 - val_loss: 4.0148 - val_accurac
y: 0.4658
Epoch 67/200
391/391 - 5s - loss: 3.9511 - accuracy: 0.4797 - val_loss: 3.9280 - val_accurac
y: 0.4698
Epoch 68/200
391/391 - 5s - loss: 3.8649 - accuracy: 0.4809 - val_loss: 3.8437 - val_accurac
y: 0.4676
Epoch 69/200
391/391 - 5s - loss: 3.7823 - accuracy: 0.4814 - val_loss: 3.7641 - val_accurac
y: 0.4670
Epoch 70/200
391/391 - 5s - loss: 3.7032 - accuracy: 0.4816 - val_loss: 3.6840 - val_accurac
y: 0.4715
Epoch 71/200
391/391 - 5s - loss: 3.6269 - accuracy: 0.4825 - val_loss: 3.6097 - val_accurac
y: 0.4744
Epoch 72/200
391/391 - 5s - loss: 3.5533 - accuracy: 0.4819 - val_loss: 3.5423 - val_accurac
y: 0.4667
Epoch 73/200
391/391 - 5s - loss: 3.4835 - accuracy: 0.4842 - val_loss: 3.4701 - val_accurac
y: 0.4698
Epoch 74/200
391/391 - 5s - loss: 3.4163 - accuracy: 0.4841 - val_loss: 3.4037 - val_accurac
y: 0.4697
Epoch 75/200
391/391 - 5s - loss: 3.3514 - accuracy: 0.4833 - val_loss: 3.3416 - val_accurac
y: 0.4732
Epoch 76/200
391/391 - 5s - loss: 3.2893 - accuracy: 0.4856 - val_loss: 3.2803 - val_accurac
y: 0.4750
Epoch 77/200
391/391 - 5s - loss: 3.2298 - accuracy: 0.4853 - val_loss: 3.2221 - val_accurac
y: 0.4731
Epoch 78/200
391/391 - 5s - loss: 3.1722 - accuracy: 0.4861 - val_loss: 3.1732 - val_accurac
y: 0.4695
Epoch 79/200
391/391 - 5s - loss: 3.1170 - accuracy: 0.4866 - val_loss: 3.1177 - val_accurac
```

```
y: 0.4685
Epoch 80/200
391/391 - 5s - loss: 3.0641 - accuracy: 0.4862 - val_loss: 3.0664 - val_accurac
y: 0.4714
Epoch 81/200
391/391 - 5s - loss: 3.0135 - accuracy: 0.4874 - val_loss: 3.0123 - val_accurac
y: 0.4768
Epoch 82/200
391/391 - 5s - loss: 2.9644 - accuracy: 0.4870 - val_loss: 2.9644 - val_accurac
y: 0.4744
Epoch 83/200
391/391 - 5s - loss: 2.9169 - accuracy: 0.4878 - val_loss: 2.9247 - val_accurac
y: 0.4745
Epoch 84/200
391/391 - 5s - loss: 2.8724 - accuracy: 0.4896 - val_loss: 2.8752 - val_accurac
y: 0.4727
Epoch 85/200
391/391 - 5s - loss: 2.8287 - accuracy: 0.4893 - val_loss: 2.8336 - val_accurac
y: 0.4784
Epoch 86/200
391/391 - 5s - loss: 2.7865 - accuracy: 0.4901 - val_loss: 2.7886 - val_accurac
y: 0.4800
Epoch 87/200
391/391 - 5s - loss: 2.7464 - accuracy: 0.4903 - val_loss: 2.7521 - val_accurac
y: 0.4770
Epoch 88/200
391/391 - 5s - loss: 2.7079 - accuracy: 0.4905 - val_loss: 2.7157 - val_accurac
y: 0.4728
Epoch 89/200
391/391 - 5s - loss: 2.6702 - accuracy: 0.4914 - val_loss: 2.6770 - val_accurac
y: 0.4799
Epoch 90/200
391/391 - 5s - loss: 2.6342 - accuracy: 0.4928 - val_loss: 2.6456 - val_accurac
y: 0.4760
Epoch 91/200
391/391 - 5s - loss: 2.6003 - accuracy: 0.4927 - val_loss: 2.6068 - val_accurac
y: 0.4813
Epoch 92/200
391/391 - 5s - loss: 2.5670 - accuracy: 0.4934 - val_loss: 2.5739 - val_accurac
y: 0.4832
Epoch 93/200
391/391 - 5s - loss: 2.5347 - accuracy: 0.4939 - val_loss: 2.5458 - val_accurac
y: 0.4786
Epoch 94/200
391/391 - 5s - loss: 2.5036 - accuracy: 0.4938 - val_loss: 2.5145 - val_accurac
y: 0.4782
Epoch 95/200
391/391 - 5s - loss: 2.4744 - accuracy: 0.4945 - val_loss: 2.4936 - val_accurac
y: 0.4769
Epoch 96/200
391/391 - 5s - loss: 2.4457 - accuracy: 0.4947 - val_loss: 2.4621 - val_accurac
y: 0.4795
Epoch 97/200
391/391 - 5s - loss: 2.4183 - accuracy: 0.4962 - val_loss: 2.4310 - val_accurac
y: 0.4802
Epoch 98/200
391/391 - 5s - loss: 2.3918 - accuracy: 0.4957 - val_loss: 2.4032 - val_accurac
y: 0.4825
Epoch 99/200
```

```
391/391 - 5s - loss: 2.3659 - accuracy: 0.4973 - val_loss: 2.3792 - val_accurac
y: 0.4817
Epoch 100/200
391/391 - 5s - loss: 2.3414 - accuracy: 0.4972 - val_loss: 2.3575 - val_accurac
y: 0.4833
Epoch 101/200
391/391 - 5s - loss: 2.3175 - accuracy: 0.4963 - val_loss: 2.3321 - val_accurac
y: 0.4832
Epoch 102/200
391/391 - 5s - loss: 2.2953 - accuracy: 0.4979 - val_loss: 2.3165 - val_accurac
y: 0.4804
Epoch 103/200
391/391 - 5s - loss: 2.2728 - accuracy: 0.4966 - val_loss: 2.2884 - val_accurac
y: 0.4834
Epoch 104/200
391/391 - 5s - loss: 2.2513 - accuracy: 0.4985 - val_loss: 2.2667 - val_accurac
y: 0.4856
Epoch 105/200
391/391 - 5s - loss: 2.2307 - accuracy: 0.4986 - val_loss: 2.2472 - val_accurac
y: 0.4833
Epoch 106/200
391/391 - 5s - loss: 2.2106 - accuracy: 0.5006 - val_loss: 2.2349 - val_accurac
y: 0.4870
Epoch 107/200
391/391 - 5s - loss: 2.1918 - accuracy: 0.5001 - val_loss: 2.2099 - val_accurac
y: 0.4893
Epoch 108/200
391/391 - 5s - loss: 2.1732 - accuracy: 0.5015 - val_loss: 2.1989 - val_accurac
y: 0.4846
Epoch 109/200
391/391 - 5s - loss: 2.1552 - accuracy: 0.5020 - val_loss: 2.1863 - val_accurac
y: 0.4861
Epoch 110/200
391/391 - 5s - loss: 2.1385 - accuracy: 0.5017 - val_loss: 2.1619 - val_accurac
y: 0.4886
Epoch 111/200
391/391 - 5s - loss: 2.1218 - accuracy: 0.5013 - val_loss: 2.1406 - val_accurac
y: 0.4887
Epoch 112/200
391/391 - 5s - loss: 2.1059 - accuracy: 0.5008 - val_loss: 2.1288 - val_accurac
y: 0.4838
Epoch 113/200
391/391 - 5s - loss: 2.0902 - accuracy: 0.5026 - val_loss: 2.1106 - val_accurac
y: 0.4864
Epoch 114/200
391/391 - 5s - loss: 2.0755 - accuracy: 0.5039 - val_loss: 2.0966 - val_accurac
y: 0.4895
Epoch 115/200
391/391 - 5s - loss: 2.0609 - accuracy: 0.5044 - val_loss: 2.0891 - val_accurac
y: 0.4871
Epoch 116/200
391/391 - 5s - loss: 2.0466 - accuracy: 0.5037 - val_loss: 2.0682 - val_accurac
y: 0.4908
Epoch 117/200
391/391 - 5s - loss: 2.0334 - accuracy: 0.5045 - val_loss: 2.0594 - val_accurac
y: 0.4885
Epoch 118/200
391/391 - 5s - loss: 2.0205 - accuracy: 0.5054 - val_loss: 2.0424 - val_accurac
y: 0.4903
```

```
Epoch 119/200
391/391 - 5s - loss: 2.0080 - accuracy: 0.5049 - val_loss: 2.0344 - val_accurac
y: 0.4884
Epoch 120/200
391/391 - 5s - loss: 1.9955 - accuracy: 0.5059 - val_loss: 2.0230 - val_accurac
y: 0.4920
Epoch 121/200
391/391 - 5s - loss: 1.9836 - accuracy: 0.5062 - val_loss: 2.0100 - val_accurac
y: 0.4879
Epoch 122/200
391/391 - 5s - loss: 1.9718 - accuracy: 0.5059 - val_loss: 1.9992 - val_accurac
y: 0.4878
Epoch 123/200
391/391 - 5s - loss: 1.9614 - accuracy: 0.5059 - val_loss: 1.9860 - val_accurac
y: 0.4921
Epoch 124/200
391/391 - 5s - loss: 1.9505 - accuracy: 0.5075 - val_loss: 1.9760 - val_accurac
y: 0.4930
Epoch 125/200
391/391 - 5s - loss: 1.9400 - accuracy: 0.5061 - val_loss: 1.9674 - val_accurac
y: 0.4896
Epoch 126/200
391/391 - 5s - loss: 1.9301 - accuracy: 0.5084 - val_loss: 1.9572 - val_accurac
y: 0.4959
Epoch 127/200
391/391 - 5s - loss: 1.9201 - accuracy: 0.5084 - val_loss: 1.9458 - val_accurac
y: 0.4972
Epoch 128/200
391/391 - 5s - loss: 1.9109 - accuracy: 0.5092 - val_loss: 1.9394 - val_accurac
y: 0.4968
Epoch 129/200
391/391 - 5s - loss: 1.9012 - accuracy: 0.5094 - val_loss: 1.9305 - val_accurac
y: 0.4946
Epoch 130/200
391/391 - 5s - loss: 1.8933 - accuracy: 0.5092 - val_loss: 1.9224 - val_accurac
y: 0.4953
Epoch 131/200
391/391 - 5s - loss: 1.8847 - accuracy: 0.5104 - val_loss: 1.9178 - val_accurac
y: 0.4915
Epoch 132/200
391/391 - 5s - loss: 1.8762 - accuracy: 0.5111 - val_loss: 1.9075 - val_accurac
y: 0.4979
Epoch 133/200
391/391 - 5s - loss: 1.8686 - accuracy: 0.5106 - val_loss: 1.9006 - val_accurac
y: 0.4980
Epoch 134/200
391/391 - 5s - loss: 1.8606 - accuracy: 0.5111 - val_loss: 1.8938 - val_accurac
y: 0.4952
Epoch 135/200
391/391 - 5s - loss: 1.8535 - accuracy: 0.5100 - val_loss: 1.8818 - val_accurac
y: 0.4958
Epoch 136/200
391/391 - 5s - loss: 1.8451 - accuracy: 0.5113 - val_loss: 1.8752 - val_accurac
y: 0.4951
Epoch 137/200
391/391 - 5s - loss: 1.8385 - accuracy: 0.5114 - val_loss: 1.8706 - val_accurac
y: 0.4967
Epoch 138/200
391/391 - 5s - loss: 1.8318 - accuracy: 0.5124 - val_loss: 1.8611 - val_accurac
```

```
y: 0.4957
Epoch 139/200
391/391 - 5s - loss: 1.8253 - accuracy: 0.5112 - val_loss: 1.8629 - val_accurac
y: 0.4904
Epoch 140/200
391/391 - 5s - loss: 1.8185 - accuracy: 0.5135 - val_loss: 1.8532 - val_accurac
y: 0.4963
Epoch 141/200
391/391 - 5s - loss: 1.8119 - accuracy: 0.5133 - val_loss: 1.8456 - val_accurac
y: 0.5000
Epoch 142/200
391/391 - 5s - loss: 1.8058 - accuracy: 0.5144 - val_loss: 1.8419 - val_accurac
y: 0.4936
Epoch 143/200
391/391 - 5s - loss: 1.8000 - accuracy: 0.5147 - val_loss: 1.8341 - val_accurac
y: 0.4979
Epoch 144/200
391/391 - 5s - loss: 1.7942 - accuracy: 0.5143 - val_loss: 1.8251 - val_accurac
y: 0.5000
Epoch 145/200
391/391 - 5s - loss: 1.7884 - accuracy: 0.5146 - val_loss: 1.8253 - val_accurac
y: 0.4978
Epoch 146/200
391/391 - 5s - loss: 1.7829 - accuracy: 0.5146 - val_loss: 1.8264 - val_accurac
y: 0.4967
Epoch 147/200
391/391 - 5s - loss: 1.7774 - accuracy: 0.5172 - val_loss: 1.8172 - val_accurac
y: 0.4987
Epoch 148/200
391/391 - 5s - loss: 1.7725 - accuracy: 0.5172 - val_loss: 1.8059 - val_accurac
y: 0.4987
Epoch 149/200
391/391 - 5s - loss: 1.7677 - accuracy: 0.5164 - val_loss: 1.8045 - val_accurac
y: 0.4958
Epoch 150/200
391/391 - 5s - loss: 1.7626 - accuracy: 0.5174 - val_loss: 1.7988 - val_accurac
y: 0.4983
Epoch 151/200
391/391 - 5s - loss: 1.7579 - accuracy: 0.5166 - val_loss: 1.7932 - val_accurac
y: 0.4963
Epoch 152/200
391/391 - 5s - loss: 1.7534 - accuracy: 0.5165 - val_loss: 1.7872 - val_accurac
y: 0.5050
Epoch 153/200
391/391 - 5s - loss: 1.7488 - accuracy: 0.5174 - val_loss: 1.7815 - val_accurac
y: 0.5040
Epoch 154/200
391/391 - 5s - loss: 1.7443 - accuracy: 0.5178 - val_loss: 1.7835 - val_accurac
y: 0.4983
Epoch 155/200
391/391 - 5s - loss: 1.7401 - accuracy: 0.5175 - val_loss: 1.7767 - val_accurac
y: 0.5000
Epoch 156/200
391/391 - 5s - loss: 1.7354 - accuracy: 0.5180 - val_loss: 1.7732 - val_accurac
y: 0.5006
Epoch 157/200
391/391 - 5s - loss: 1.7320 - accuracy: 0.5182 - val_loss: 1.7742 - val_accurac
y: 0.4905
Epoch 158/200
```

```
391/391 - 5s - loss: 1.7277 - accuracy: 0.5193 - val_loss: 1.7691 - val_accurac
y: 0.5010
Epoch 159/200
391/391 - 5s - loss: 1.7242 - accuracy: 0.5196 - val_loss: 1.7658 - val_accurac
y: 0.4978
Epoch 160/200
391/391 - 5s - loss: 1.7201 - accuracy: 0.5191 - val_loss: 1.7596 - val_accurac
y: 0.5048
Epoch 161/200
391/391 - 5s - loss: 1.7169 - accuracy: 0.5193 - val_loss: 1.7596 - val_accurac
y: 0.4980
Epoch 162/200
391/391 - 5s - loss: 1.7128 - accuracy: 0.5188 - val_loss: 1.7685 - val_accurac
y: 0.4930
Epoch 163/200
391/391 - 5s - loss: 1.7096 - accuracy: 0.5206 - val_loss: 1.7454 - val_accurac
y: 0.5053
Epoch 164/200
391/391 - 5s - loss: 1.7064 - accuracy: 0.5196 - val_loss: 1.7473 - val_accurac
y: 0.5015
Epoch 165/200
391/391 - 5s - loss: 1.7031 - accuracy: 0.5203 - val_loss: 1.7449 - val_accurac
y: 0.5019
Epoch 166/200
391/391 - 5s - loss: 1.6996 - accuracy: 0.5212 - val_loss: 1.7397 - val_accurac
y: 0.5007
Epoch 167/200
391/391 - 5s - loss: 1.6963 - accuracy: 0.5214 - val_loss: 1.7372 - val_accurac
y: 0.5013
Epoch 168/200
391/391 - 5s - loss: 1.6933 - accuracy: 0.5219 - val_loss: 1.7322 - val_accurac
y: 0.5053
Epoch 169/200
391/391 - 5s - loss: 1.6904 - accuracy: 0.5215 - val_loss: 1.7313 - val_accurac
y: 0.4976
Epoch 170/200
391/391 - 5s - loss: 1.6876 - accuracy: 0.5226 - val_loss: 1.7285 - val_accurac
y: 0.5050
Epoch 171/200
391/391 - 5s - loss: 1.6843 - accuracy: 0.5229 - val_loss: 1.7258 - val_accurac
y: 0.5069
Epoch 172/200
391/391 - 5s - loss: 1.6820 - accuracy: 0.5230 - val_loss: 1.7276 - val_accurac
y: 0.4990
Epoch 173/200
391/391 - 5s - loss: 1.6794 - accuracy: 0.5220 - val_loss: 1.7236 - val_accurac
y: 0.5027
Epoch 174/200
391/391 - 5s - loss: 1.6762 - accuracy: 0.5233 - val_loss: 1.7145 - val_accurac
y: 0.5063
Epoch 175/200
391/391 - 5s - loss: 1.6740 - accuracy: 0.5220 - val_loss: 1.7201 - val_accurac
y: 0.5010
Epoch 176/200
391/391 - 5s - loss: 1.6709 - accuracy: 0.5239 - val_loss: 1.7131 - val_accurac
y: 0.5031
Epoch 177/200
391/391 - 5s - loss: 1.6688 - accuracy: 0.5250 - val_loss: 1.7197 - val_accurac
y: 0.5014
```

```
Epoch 178/200
391/391 - 5s - loss: 1.6664 - accuracy: 0.5237 - val_loss: 1.7074 - val_accurac
y: 0.5024
Epoch 179/200
391/391 - 5s - loss: 1.6640 - accuracy: 0.5240 - val_loss: 1.7077 - val_accurac
y: 0.5051
Epoch 180/200
391/391 - 5s - loss: 1.6612 - accuracy: 0.5256 - val_loss: 1.7102 - val_accurac
y: 0.5002
Epoch 181/200
391/391 - 5s - loss: 1.6590 - accuracy: 0.5239 - val_loss: 1.7007 - val_accurac
y: 0.5056
Epoch 182/200
391/391 - 5s - loss: 1.6572 - accuracy: 0.5253 - val_loss: 1.7024 - val_accurac
y: 0.5028
Epoch 183/200
391/391 - 5s - loss: 1.6551 - accuracy: 0.5249 - val_loss: 1.6983 - val_accurac
y: 0.5070
Epoch 184/200
391/391 - 5s - loss: 1.6527 - accuracy: 0.5268 - val_loss: 1.6970 - val_accurac
y: 0.5103
Epoch 185/200
391/391 - 5s - loss: 1.6506 - accuracy: 0.5258 - val_loss: 1.6986 - val_accurac
y: 0.5048
Epoch 186/200
391/391 - 5s - loss: 1.6484 - accuracy: 0.5259 - val_loss: 1.6927 - val_accurac
y: 0.5055
Epoch 187/200
391/391 - 5s - loss: 1.6465 - accuracy: 0.5268 - val_loss: 1.6966 - val_accurac
y: 0.4989
Epoch 188/200
391/391 - 5s - loss: 1.6443 - accuracy: 0.5271 - val_loss: 1.6984 - val_accurac
y: 0.5033
Epoch 189/200
391/391 - 5s - loss: 1.6425 - accuracy: 0.5278 - val_loss: 1.6908 - val_accurac
y: 0.5051
Epoch 190/200
391/391 - 5s - loss: 1.6406 - accuracy: 0.5277 - val_loss: 1.6825 - val_accurac
y: 0.5118
Epoch 191/200
391/391 - 5s - loss: 1.6386 - accuracy: 0.5273 - val_loss: 1.6815 - val_accurac
y: 0.5103
Epoch 192/200
391/391 - 5s - loss: 1.6369 - accuracy: 0.5274 - val_loss: 1.6798 - val_accurac
y: 0.5129
Epoch 193/200
391/391 - 5s - loss: 1.6350 - accuracy: 0.5285 - val_loss: 1.6785 - val_accurac
y: 0.5093
Epoch 194/200
391/391 - 5s - loss: 1.6332 - accuracy: 0.5294 - val_loss: 1.6861 - val_accurac
y: 0.5034
Epoch 195/200
391/391 - 5s - loss: 1.6317 - accuracy: 0.5298 - val_loss: 1.6777 - val_accurac
y: 0.5097
Epoch 196/200
391/391 - 5s - loss: 1.6299 - accuracy: 0.5286 - val_loss: 1.6755 - val_accurac
y: 0.5091
Epoch 197/200
391/391 - 5s - loss: 1.6281 - accuracy: 0.5291 - val_loss: 1.6820 - val_accurac
```

```
y: 0.5002
Epoch 198/200
391/391 - 5s - loss: 1.6267 - accuracy: 0.5291 - val_loss: 1.6835 - val_accurac
y: 0.5047
Epoch 199/200
391/391 - 5s - loss: 1.6250 - accuracy: 0.5301 - val_loss: 1.6704 - val_accurac
y: 0.5085
Epoch 200/200
391/391 - 5s - loss: 1.6235 - accuracy: 0.5289 - val_loss: 1.6799 - val_accurac
y: 0.5031
```

RMSProp

In [18]:
```
model_rmsprop=create_model()
model_rmsprop.compile(loss='categorical_crossentropy', optimizer='RMSprop', metri
cs=['accuracy'])
history_rmsprop = model_rmsprop.fit(X_train, y_train, batch_size=128, epochs=200,
validation_data=(X_test,y_test),shuffle=True,verbose=2)
```

```
Epoch 1/200
391/391 - 8s - loss: 5.8721 - accuracy: 0.2145 - val_loss: 2.2389 - val_accurac
y: 0.2267
Epoch 2/200
391/391 - 7s - loss: 2.1274 - accuracy: 0.2810 - val_loss: 2.0811 - val_accurac
y: 0.2940
Epoch 3/200
391/391 - 7s - loss: 2.0895 - accuracy: 0.2959 - val_loss: 1.9812 - val_accurac
y: 0.3316
Epoch 4/200
391/391 - 6s - loss: 2.0476 - accuracy: 0.3086 - val_loss: 1.9727 - val_accurac
y: 0.3327
Epoch 5/200
391/391 - 6s - loss: 2.0128 - accuracy: 0.3203 - val_loss: 2.1070 - val_accurac
y: 0.2918
Epoch 6/200
391/391 - 6s - loss: 1.9773 - accuracy: 0.3309 - val_loss: 2.0508 - val_accurac
y: 0.3032
Epoch 7/200
391/391 - 6s - loss: 1.9504 - accuracy: 0.3355 - val_loss: 1.9768 - val_accurac
y: 0.3239
Epoch 8/200
391/391 - 6s - loss: 1.9325 - accuracy: 0.3475 - val_loss: 1.9518 - val_accurac
y: 0.3281
Epoch 9/200
391/391 - 6s - loss: 1.9162 - accuracy: 0.3478 - val_loss: 1.9292 - val_accurac
y: 0.3329
Epoch 10/200
391/391 - 6s - loss: 1.9072 - accuracy: 0.3487 - val_loss: 1.8945 - val_accurac
y: 0.3500
Epoch 11/200
391/391 - 6s - loss: 1.8966 - accuracy: 0.3548 - val_loss: 1.8860 - val_accurac
y: 0.3586
Epoch 12/200
391/391 - 6s - loss: 1.8921 - accuracy: 0.3546 - val_loss: 1.8651 - val_accurac
y: 0.3630
Epoch 13/200
391/391 - 6s - loss: 1.8902 - accuracy: 0.3556 - val_loss: 1.9252 - val_accurac
y: 0.3393
Epoch 14/200
391/391 - 6s - loss: 1.8876 - accuracy: 0.3566 - val_loss: 1.9942 - val_accurac
y: 0.3203
Epoch 15/200
391/391 - 6s - loss: 1.8844 - accuracy: 0.3589 - val_loss: 1.9776 - val_accurac
y: 0.3228
Epoch 16/200
391/391 - 6s - loss: 1.8839 - accuracy: 0.3620 - val_loss: 1.8062 - val_accurac
y: 0.3960
Epoch 17/200
391/391 - 6s - loss: 1.8823 - accuracy: 0.3623 - val_loss: 1.8070 - val_accurac
y: 0.3865
Epoch 18/200
391/391 - 7s - loss: 1.8760 - accuracy: 0.3606 - val_loss: 1.8992 - val_accurac
y: 0.3452
Epoch 19/200
391/391 - 6s - loss: 1.8797 - accuracy: 0.3615 - val_loss: 1.9517 - val_accurac
y: 0.3387
Epoch 20/200
391/391 - 6s - loss: 1.8759 - accuracy: 0.3618 - val_loss: 1.8642 - val_accurac
```

```
y: 0.3541
Epoch 21/200
391/391 - 6s - loss: 1.8755 - accuracy: 0.3617 - val_loss: 1.9197 - val_accurac
y: 0.3491
Epoch 22/200
391/391 - 6s - loss: 1.8761 - accuracy: 0.3599 - val_loss: 1.9718 - val_accurac
y: 0.3339
Epoch 23/200
391/391 - 6s - loss: 1.8726 - accuracy: 0.3636 - val_loss: 1.8795 - val_accurac
y: 0.3617
Epoch 24/200
391/391 - 6s - loss: 1.8719 - accuracy: 0.3634 - val_loss: 1.8486 - val_accurac
y: 0.3709
Epoch 25/200
391/391 - 6s - loss: 1.8716 - accuracy: 0.3641 - val_loss: 1.8776 - val_accurac
y: 0.3677
Epoch 26/200
391/391 - 6s - loss: 1.8723 - accuracy: 0.3649 - val_loss: 1.8561 - val_accurac
y: 0.3711
Epoch 27/200
391/391 - 7s - loss: 1.8683 - accuracy: 0.3640 - val_loss: 1.9119 - val_accurac
y: 0.3621
Epoch 28/200
391/391 - 7s - loss: 1.8683 - accuracy: 0.3655 - val_loss: 2.0623 - val_accurac
y: 0.2956
Epoch 29/200
391/391 - 6s - loss: 1.8692 - accuracy: 0.3656 - val_loss: 1.8879 - val_accurac
y: 0.3477
Epoch 30/200
391/391 - 6s - loss: 1.8695 - accuracy: 0.3628 - val_loss: 1.8148 - val_accurac
y: 0.3895
Epoch 31/200
391/391 - 6s - loss: 1.8681 - accuracy: 0.3660 - val_loss: 1.8612 - val_accurac
y: 0.3708
Epoch 32/200
391/391 - 6s - loss: 1.8650 - accuracy: 0.3666 - val_loss: 1.9102 - val_accurac
y: 0.3657
Epoch 33/200
391/391 - 6s - loss: 1.8657 - accuracy: 0.3653 - val_loss: 1.8396 - val_accurac
y: 0.3827
Epoch 34/200
391/391 - 7s - loss: 1.8662 - accuracy: 0.3653 - val_loss: 1.9121 - val_accurac
y: 0.3529
Epoch 35/200
391/391 - 7s - loss: 1.8629 - accuracy: 0.3684 - val_loss: 1.9434 - val_accurac
y: 0.3576
Epoch 36/200
391/391 - 6s - loss: 1.8629 - accuracy: 0.3671 - val_loss: 2.1001 - val_accurac
y: 0.3035
Epoch 37/200
391/391 - 6s - loss: 1.8624 - accuracy: 0.3661 - val_loss: 1.8857 - val_accurac
y: 0.3521
Epoch 38/200
391/391 - 7s - loss: 1.8613 - accuracy: 0.3681 - val_loss: 1.8582 - val_accurac
y: 0.3724
Epoch 39/200
391/391 - 7s - loss: 1.8623 - accuracy: 0.3659 - val_loss: 1.9035 - val_accurac
y: 0.3461
Epoch 40/200
```

```
391/391 - 6s - loss: 1.8651 - accuracy: 0.3651 - val_loss: 1.8554 - val_accurac
y: 0.3646
Epoch 41/200
391/391 - 6s - loss: 1.8625 - accuracy: 0.3660 - val_loss: 1.8171 - val_accurac
y: 0.3834
Epoch 42/200
391/391 - 6s - loss: 1.8596 - accuracy: 0.3680 - val_loss: 1.9447 - val_accurac
y: 0.3465
Epoch 43/200
391/391 - 7s - loss: 1.8626 - accuracy: 0.3688 - val_loss: 1.7939 - val_accurac
y: 0.3887
Epoch 44/200
391/391 - 6s - loss: 1.8601 - accuracy: 0.3676 - val_loss: 1.8846 - val_accurac
y: 0.3597
Epoch 45/200
391/391 - 7s - loss: 1.8612 - accuracy: 0.3680 - val_loss: 2.2746 - val_accurac
y: 0.2924
Epoch 46/200
391/391 - 7s - loss: 1.8583 - accuracy: 0.3681 - val_loss: 1.8691 - val_accurac
y: 0.3536
Epoch 47/200
391/391 - 6s - loss: 1.8582 - accuracy: 0.3682 - val_loss: 1.9115 - val_accurac
y: 0.3545
Epoch 48/200
391/391 - 6s - loss: 1.8634 - accuracy: 0.3661 - val_loss: 2.0361 - val_accurac
y: 0.3147
Epoch 49/200
391/391 - 6s - loss: 1.8574 - accuracy: 0.3673 - val_loss: 1.8570 - val_accurac
y: 0.3626
Epoch 50/200
391/391 - 6s - loss: 1.8584 - accuracy: 0.3662 - val_loss: 1.8200 - val_accurac
y: 0.3756
Epoch 51/200
391/391 - 6s - loss: 1.8619 - accuracy: 0.3669 - val_loss: 1.8776 - val_accurac
y: 0.3584
Epoch 52/200
391/391 - 7s - loss: 1.8594 - accuracy: 0.3669 - val_loss: 1.8514 - val_accurac
y: 0.3692
Epoch 53/200
391/391 - 7s - loss: 1.8574 - accuracy: 0.3706 - val_loss: 1.8159 - val_accurac
y: 0.3853
Epoch 54/200
391/391 - 6s - loss: 1.8568 - accuracy: 0.3684 - val_loss: 1.8513 - val_accurac
y: 0.3727
Epoch 55/200
391/391 - 6s - loss: 1.8569 - accuracy: 0.3690 - val_loss: 1.8145 - val_accurac
y: 0.3808
Epoch 56/200
391/391 - 6s - loss: 1.8550 - accuracy: 0.3684 - val_loss: 1.9907 - val_accurac
y: 0.3098
Epoch 57/200
391/391 - 6s - loss: 1.8566 - accuracy: 0.3702 - val_loss: 1.8575 - val_accurac
y: 0.3666
Epoch 58/200
391/391 - 7s - loss: 1.8520 - accuracy: 0.3693 - val_loss: 1.8648 - val_accurac
y: 0.3576
Epoch 59/200
391/391 - 7s - loss: 1.8545 - accuracy: 0.3670 - val_loss: 1.8043 - val_accurac
y: 0.3896
```

```
Epoch 60/200
391/391 - 7s - loss: 1.8584 - accuracy: 0.3692 - val_loss: 1.8612 - val_accurac
y: 0.3737
Epoch 61/200
391/391 - 7s - loss: 1.8574 - accuracy: 0.3696 - val_loss: 1.9029 - val_accurac
y: 0.3599
Epoch 62/200
391/391 - 7s - loss: 1.8544 - accuracy: 0.3671 - val_loss: 1.8673 - val_accurac
y: 0.3669
Epoch 63/200
391/391 - 7s - loss: 1.8550 - accuracy: 0.3704 - val_loss: 1.7927 - val_accurac
y: 0.3837
Epoch 64/200
391/391 - 7s - loss: 1.8584 - accuracy: 0.3664 - val_loss: 1.8153 - val_accurac
y: 0.3853
Epoch 65/200
391/391 - 7s - loss: 1.8546 - accuracy: 0.3686 - val_loss: 1.8003 - val_accurac
y: 0.3861
Epoch 66/200
391/391 - 7s - loss: 1.8552 - accuracy: 0.3689 - val_loss: 1.9713 - val_accurac
y: 0.3370
Epoch 67/200
391/391 - 7s - loss: 1.8545 - accuracy: 0.3699 - val_loss: 1.8605 - val_accurac
y: 0.3659
Epoch 68/200
391/391 - 6s - loss: 1.8541 - accuracy: 0.3678 - val_loss: 1.8393 - val_accurac
y: 0.3714
Epoch 69/200
391/391 - 7s - loss: 1.8538 - accuracy: 0.3684 - val_loss: 1.9115 - val_accurac
y: 0.3513
Epoch 70/200
391/391 - 7s - loss: 1.8515 - accuracy: 0.3708 - val_loss: 2.1978 - val_accurac
y: 0.2928
Epoch 71/200
391/391 - 7s - loss: 1.8571 - accuracy: 0.3693 - val_loss: 1.9439 - val_accurac
y: 0.3336
Epoch 72/200
391/391 - 7s - loss: 1.8496 - accuracy: 0.3698 - val_loss: 1.7856 - val_accurac
y: 0.4014
Epoch 73/200
391/391 - 7s - loss: 1.8515 - accuracy: 0.3695 - val_loss: 1.8217 - val_accurac
y: 0.3792
Epoch 74/200
391/391 - 7s - loss: 1.8507 - accuracy: 0.3732 - val_loss: 2.0652 - val_accurac
y: 0.3078
Epoch 75/200
391/391 - 7s - loss: 1.8491 - accuracy: 0.3713 - val_loss: 1.8878 - val_accurac
y: 0.3573
Epoch 76/200
391/391 - 7s - loss: 1.8500 - accuracy: 0.3703 - val_loss: 1.8001 - val_accurac
y: 0.3914
Epoch 77/200
391/391 - 7s - loss: 1.8518 - accuracy: 0.3683 - val_loss: 1.9400 - val_accurac
y: 0.3334
Epoch 78/200
391/391 - 7s - loss: 1.8452 - accuracy: 0.3728 - val_loss: 1.8350 - val_accurac
y: 0.3715
Epoch 79/200
391/391 - 7s - loss: 1.8462 - accuracy: 0.3734 - val_loss: 1.8567 - val_accurac
```

```
y: 0.3555
Epoch 80/200
391/391 - 7s - loss: 1.8532 - accuracy: 0.3691 - val_loss: 1.8643 - val_accurac
y: 0.3622
Epoch 81/200
391/391 - 7s - loss: 1.8511 - accuracy: 0.3689 - val_loss: 1.8310 - val_accurac
y: 0.3765
Epoch 82/200
391/391 - 7s - loss: 1.8512 - accuracy: 0.3705 - val_loss: 2.3990 - val_accurac
y: 0.2617
Epoch 83/200
391/391 - 7s - loss: 1.8479 - accuracy: 0.3734 - val_loss: 1.8972 - val_accurac
y: 0.3575
Epoch 84/200
391/391 - 7s - loss: 1.8456 - accuracy: 0.3727 - val_loss: 1.8177 - val_accurac
y: 0.3820
Epoch 85/200
391/391 - 7s - loss: 1.8469 - accuracy: 0.3723 - val_loss: 1.8985 - val_accurac
y: 0.3590
Epoch 86/200
391/391 - 7s - loss: 1.8476 - accuracy: 0.3691 - val_loss: 1.8597 - val_accurac
y: 0.3728
Epoch 87/200
391/391 - 7s - loss: 1.8490 - accuracy: 0.3681 - val_loss: 1.8254 - val_accurac
y: 0.3747
Epoch 88/200
391/391 - 7s - loss: 1.8496 - accuracy: 0.3690 - val_loss: 1.8887 - val_accurac
y: 0.3449
Epoch 89/200
391/391 - 7s - loss: 1.8487 - accuracy: 0.3703 - val_loss: 1.8865 - val_accurac
y: 0.3530
Epoch 90/200
391/391 - 7s - loss: 1.8458 - accuracy: 0.3723 - val_loss: 1.8473 - val_accurac
y: 0.3725
Epoch 91/200
391/391 - 7s - loss: 1.8448 - accuracy: 0.3726 - val_loss: 1.8256 - val_accurac
y: 0.3883
Epoch 92/200
391/391 - 7s - loss: 1.8474 - accuracy: 0.3705 - val_loss: 1.7764 - val_accurac
y: 0.3975
Epoch 93/200
391/391 - 7s - loss: 1.8456 - accuracy: 0.3717 - val_loss: 1.9479 - val_accurac
y: 0.3365
Epoch 94/200
391/391 - 7s - loss: 1.8478 - accuracy: 0.3697 - val_loss: 1.9190 - val_accurac
y: 0.3516
Epoch 95/200
391/391 - 7s - loss: 1.8493 - accuracy: 0.3714 - val_loss: 1.7761 - val_accurac
y: 0.4037
Epoch 96/200
391/391 - 7s - loss: 1.8436 - accuracy: 0.3737 - val_loss: 1.9293 - val_accurac
y: 0.3325
Epoch 97/200
391/391 - 7s - loss: 1.8487 - accuracy: 0.3736 - val_loss: 1.8351 - val_accurac
y: 0.3765
Epoch 98/200
391/391 - 7s - loss: 1.8472 - accuracy: 0.3729 - val_loss: 1.9232 - val_accurac
y: 0.3469
Epoch 99/200
```

```
391/391 - 7s - loss: 1.8455 - accuracy: 0.3716 - val_loss: 1.8438 - val_accurac
y: 0.3653
Epoch 100/200
391/391 - 7s - loss: 1.8432 - accuracy: 0.3725 - val_loss: 1.7717 - val_accurac
y: 0.4025
Epoch 101/200
391/391 - 7s - loss: 1.8437 - accuracy: 0.3743 - val_loss: 1.9165 - val_accurac
y: 0.3600
Epoch 102/200
391/391 - 7s - loss: 1.8473 - accuracy: 0.3751 - val_loss: 1.9501 - val_accurac
y: 0.3245
Epoch 103/200
391/391 - 7s - loss: 1.8412 - accuracy: 0.3738 - val_loss: 2.0626 - val_accurac
y: 0.2977
Epoch 104/200
391/391 - 7s - loss: 1.8444 - accuracy: 0.3744 - val_loss: 1.8023 - val_accurac
y: 0.3854
Epoch 105/200
391/391 - 6s - loss: 1.8384 - accuracy: 0.3735 - val_loss: 1.8570 - val_accurac
y: 0.3743
Epoch 106/200
391/391 - 7s - loss: 1.8446 - accuracy: 0.3739 - val_loss: 1.8565 - val_accurac
y: 0.3612
Epoch 107/200
391/391 - 7s - loss: 1.8448 - accuracy: 0.3727 - val_loss: 1.8446 - val_accurac
y: 0.3794
Epoch 108/200
391/391 - 7s - loss: 1.8416 - accuracy: 0.3750 - val_loss: 1.7594 - val_accurac
y: 0.4085
Epoch 109/200
391/391 - 7s - loss: 1.8457 - accuracy: 0.3721 - val_loss: 1.8461 - val_accurac
y: 0.3665
Epoch 110/200
391/391 - 6s - loss: 1.8421 - accuracy: 0.3738 - val_loss: 1.8472 - val_accurac
y: 0.3760
Epoch 111/200
391/391 - 6s - loss: 1.8388 - accuracy: 0.3741 - val_loss: 1.9403 - val_accurac
y: 0.3470
Epoch 112/200
391/391 - 6s - loss: 1.8457 - accuracy: 0.3749 - val_loss: 1.8053 - val_accurac
y: 0.3968
Epoch 113/200
391/391 - 7s - loss: 1.8415 - accuracy: 0.3737 - val_loss: 1.7899 - val_accurac
y: 0.3959
Epoch 114/200
391/391 - 7s - loss: 1.8422 - accuracy: 0.3741 - val_loss: 1.8053 - val_accurac
y: 0.3963
Epoch 115/200
391/391 - 7s - loss: 1.8429 - accuracy: 0.3740 - val_loss: 1.7996 - val_accurac
y: 0.3889
Epoch 116/200
391/391 - 7s - loss: 1.8373 - accuracy: 0.3751 - val_loss: 1.8637 - val_accurac
y: 0.3589
Epoch 117/200
391/391 - 7s - loss: 1.8391 - accuracy: 0.3762 - val_loss: 1.8604 - val_accurac
y: 0.3780
Epoch 118/200
391/391 - 7s - loss: 1.8411 - accuracy: 0.3740 - val_loss: 1.7878 - val_accurac
y: 0.3896
```

```
Epoch 119/200
391/391 - 6s - loss: 1.8388 - accuracy: 0.3721 - val_loss: 1.9712 - val_accurac
y: 0.3475
Epoch 120/200
391/391 - 7s - loss: 1.8396 - accuracy: 0.3742 - val_loss: 1.7632 - val_accurac
y: 0.4056
Epoch 121/200
391/391 - 7s - loss: 1.8424 - accuracy: 0.3753 - val_loss: 1.8430 - val_accurac
y: 0.3632
Epoch 122/200
391/391 - 7s - loss: 1.8358 - accuracy: 0.3766 - val_loss: 1.8465 - val_accurac
y: 0.3676
Epoch 123/200
391/391 - 7s - loss: 1.8380 - accuracy: 0.3749 - val_loss: 1.8725 - val_accurac
y: 0.3633
Epoch 124/200
391/391 - 7s - loss: 1.8366 - accuracy: 0.3777 - val_loss: 1.7279 - val_accurac
y: 0.4191
Epoch 125/200
391/391 - 7s - loss: 1.8386 - accuracy: 0.3766 - val_loss: 1.9888 - val_accurac
y: 0.3255
Epoch 126/200
391/391 - 7s - loss: 1.8421 - accuracy: 0.3779 - val_loss: 1.8948 - val_accurac
y: 0.3567
Epoch 127/200
391/391 - 7s - loss: 1.8362 - accuracy: 0.3759 - val_loss: 1.8451 - val_accurac
y: 0.3777
Epoch 128/200
391/391 - 7s - loss: 1.8381 - accuracy: 0.3764 - val_loss: 1.9165 - val_accurac
y: 0.3506
Epoch 129/200
391/391 - 6s - loss: 1.8378 - accuracy: 0.3765 - val_loss: 1.8168 - val_accurac
y: 0.3816
Epoch 130/200
391/391 - 7s - loss: 1.8379 - accuracy: 0.3745 - val_loss: 2.0582 - val_accurac
y: 0.3221
Epoch 131/200
391/391 - 7s - loss: 1.8409 - accuracy: 0.3756 - val_loss: 1.8195 - val_accurac
y: 0.3795
Epoch 132/200
391/391 - 7s - loss: 1.8368 - accuracy: 0.3762 - val_loss: 1.8823 - val_accurac
y: 0.3513
Epoch 133/200
391/391 - 7s - loss: 1.8359 - accuracy: 0.3746 - val_loss: 2.0063 - val_accurac
y: 0.3189
Epoch 134/200
391/391 - 7s - loss: 1.8381 - accuracy: 0.3765 - val_loss: 1.8385 - val_accurac
y: 0.3695
Epoch 135/200
391/391 - 7s - loss: 1.8393 - accuracy: 0.3769 - val_loss: 1.9275 - val_accurac
y: 0.3516
Epoch 136/200
391/391 - 7s - loss: 1.8337 - accuracy: 0.3761 - val_loss: 1.8040 - val_accurac
y: 0.3943
Epoch 137/200
391/391 - 7s - loss: 1.8348 - accuracy: 0.3762 - val_loss: 1.8521 - val_accurac
y: 0.3646
Epoch 138/200
391/391 - 7s - loss: 1.8374 - accuracy: 0.3765 - val_loss: 1.8822 - val_accurac
```

```
y: 0.3571
Epoch 139/200
391/391 - 7s - loss: 1.8335 - accuracy: 0.3770 - val_loss: 1.8762 - val_accurac
y: 0.3540
Epoch 140/200
391/391 - 7s - loss: 1.8402 - accuracy: 0.3758 - val_loss: 1.8748 - val_accurac
y: 0.3626
Epoch 141/200
391/391 - 7s - loss: 1.8369 - accuracy: 0.3782 - val_loss: 2.0428 - val_accurac
y: 0.3142
Epoch 142/200
391/391 - 7s - loss: 1.8362 - accuracy: 0.3761 - val_loss: 1.7791 - val_accurac
y: 0.3970
Epoch 143/200
391/391 - 7s - loss: 1.8389 - accuracy: 0.3759 - val_loss: 1.8009 - val_accurac
y: 0.4025
Epoch 144/200
391/391 - 7s - loss: 1.8405 - accuracy: 0.3756 - val_loss: 1.8589 - val_accurac
y: 0.3673
Epoch 145/200
391/391 - 7s - loss: 1.8371 - accuracy: 0.3752 - val_loss: 1.8160 - val_accurac
y: 0.3766
Epoch 146/200
391/391 - 7s - loss: 1.8351 - accuracy: 0.3782 - val_loss: 1.8775 - val_accurac
y: 0.3589
Epoch 147/200
391/391 - 7s - loss: 1.8347 - accuracy: 0.3775 - val_loss: 2.1872 - val_accurac
y: 0.2978
Epoch 148/200
391/391 - 7s - loss: 1.8327 - accuracy: 0.3803 - val_loss: 1.7759 - val_accurac
y: 0.4041
Epoch 149/200
391/391 - 7s - loss: 1.8362 - accuracy: 0.3761 - val_loss: 1.8871 - val_accurac
y: 0.3560
Epoch 150/200
391/391 - 7s - loss: 1.8356 - accuracy: 0.3744 - val_loss: 2.0612 - val_accurac
y: 0.3141
Epoch 151/200
391/391 - 7s - loss: 1.8401 - accuracy: 0.3771 - val_loss: 1.7765 - val_accurac
y: 0.3985
Epoch 152/200
391/391 - 7s - loss: 1.8406 - accuracy: 0.3764 - val_loss: 1.8940 - val_accurac
y: 0.3583
Epoch 153/200
391/391 - 7s - loss: 1.8364 - accuracy: 0.3785 - val_loss: 1.7631 - val_accurac
y: 0.4050
Epoch 154/200
391/391 - 7s - loss: 1.8334 - accuracy: 0.3795 - val_loss: 1.8144 - val_accurac
y: 0.3884
Epoch 155/200
391/391 - 7s - loss: 1.8332 - accuracy: 0.3781 - val_loss: 1.7995 - val_accurac
y: 0.3931
Epoch 156/200
391/391 - 7s - loss: 1.8319 - accuracy: 0.3792 - val_loss: 2.0044 - val_accurac
y: 0.3263
Epoch 157/200
391/391 - 7s - loss: 1.8341 - accuracy: 0.3761 - val_loss: 1.9227 - val_accurac
y: 0.3436
Epoch 158/200
```

```
391/391 - 7s - loss: 1.8377 - accuracy: 0.3757 - val_loss: 1.8230 - val_accurac
y: 0.3847
Epoch 159/200
391/391 - 7s - loss: 1.8358 - accuracy: 0.3798 - val_loss: 1.7181 - val_accurac
y: 0.4190
Epoch 160/200
391/391 - 7s - loss: 1.8357 - accuracy: 0.3758 - val_loss: 1.9146 - val_accurac
y: 0.3575
Epoch 161/200
391/391 - 7s - loss: 1.8369 - accuracy: 0.3779 - val_loss: 1.7361 - val_accurac
y: 0.4203
Epoch 162/200
391/391 - 7s - loss: 1.8339 - accuracy: 0.3805 - val_loss: 1.8408 - val_accurac
y: 0.3701
Epoch 163/200
391/391 - 7s - loss: 1.8374 - accuracy: 0.3775 - val_loss: 1.8695 - val_accurac
y: 0.3686
Epoch 164/200
391/391 - 6s - loss: 1.8331 - accuracy: 0.3788 - val_loss: 1.8211 - val_accurac
y: 0.3841
Epoch 165/200
391/391 - 6s - loss: 1.8344 - accuracy: 0.3777 - val_loss: 1.8827 - val_accurac
y: 0.3536
Epoch 166/200
391/391 - 7s - loss: 1.8342 - accuracy: 0.3766 - val_loss: 1.7708 - val_accurac
y: 0.3969
Epoch 167/200
391/391 - 7s - loss: 1.8393 - accuracy: 0.3762 - val_loss: 1.8009 - val_accurac
y: 0.3837
Epoch 168/200
391/391 - 7s - loss: 1.8334 - accuracy: 0.3762 - val_loss: 1.8373 - val_accurac
y: 0.3708
Epoch 169/200
391/391 - 7s - loss: 1.8382 - accuracy: 0.3754 - val_loss: 1.8037 - val_accurac
y: 0.3988
Epoch 170/200
391/391 - 7s - loss: 1.8388 - accuracy: 0.3753 - val_loss: 1.9524 - val_accurac
y: 0.3428
Epoch 171/200
391/391 - 7s - loss: 1.8310 - accuracy: 0.3780 - val_loss: 2.1022 - val_accurac
y: 0.2947
Epoch 172/200
391/391 - 7s - loss: 1.8384 - accuracy: 0.3751 - val_loss: 1.8111 - val_accurac
y: 0.3844
Epoch 173/200
391/391 - 7s - loss: 1.8363 - accuracy: 0.3797 - val_loss: 1.8164 - val_accurac
y: 0.3837
Epoch 174/200
391/391 - 7s - loss: 1.8369 - accuracy: 0.3784 - val_loss: 1.7911 - val_accurac
y: 0.3926
Epoch 175/200
391/391 - 7s - loss: 1.8351 - accuracy: 0.3789 - val_loss: 1.7901 - val_accurac
y: 0.3960
Epoch 176/200
391/391 - 7s - loss: 1.8325 - accuracy: 0.3781 - val_loss: 1.9747 - val_accurac
y: 0.3140
Epoch 177/200
391/391 - 7s - loss: 1.8374 - accuracy: 0.3793 - val_loss: 1.7776 - val_accurac
y: 0.3971
```

```
Epoch 178/200
391/391 - 6s - loss: 1.8372 - accuracy: 0.3768 - val_loss: 1.8455 - val_accurac
y: 0.3765
Epoch 179/200
391/391 - 7s - loss: 1.8376 - accuracy: 0.3795 - val_loss: 1.8141 - val_accurac
y: 0.3835
Epoch 180/200
391/391 - 7s - loss: 1.8346 - accuracy: 0.3779 - val_loss: 1.7926 - val_accurac
y: 0.3993
Epoch 181/200
391/391 - 7s - loss: 1.8321 - accuracy: 0.3777 - val_loss: 1.8137 - val_accurac
y: 0.3888
Epoch 182/200
391/391 - 7s - loss: 1.8302 - accuracy: 0.3812 - val_loss: 2.0291 - val_accurac
y: 0.3099
Epoch 183/200
391/391 - 7s - loss: 1.8349 - accuracy: 0.3787 - val_loss: 1.9375 - val_accurac
y: 0.3395
Epoch 184/200
391/391 - 7s - loss: 1.8312 - accuracy: 0.3762 - val_loss: 1.8133 - val_accurac
y: 0.3793
Epoch 185/200
391/391 - 7s - loss: 1.8339 - accuracy: 0.3767 - val_loss: 1.7701 - val_accurac
y: 0.4089
Epoch 186/200
391/391 - 7s - loss: 1.8322 - accuracy: 0.3768 - val_loss: 1.8909 - val_accurac
y: 0.3631
Epoch 187/200
391/391 - 7s - loss: 1.8329 - accuracy: 0.3783 - val_loss: 1.7742 - val_accurac
y: 0.4029
Epoch 188/200
391/391 - 7s - loss: 1.8347 - accuracy: 0.3767 - val_loss: 1.7883 - val_accurac
y: 0.3988
Epoch 189/200
391/391 - 7s - loss: 1.8307 - accuracy: 0.3780 - val_loss: 1.7846 - val_accurac
y: 0.3937
Epoch 190/200
391/391 - 7s - loss: 1.8363 - accuracy: 0.3787 - val_loss: 1.8686 - val_accurac
y: 0.3609
Epoch 191/200
391/391 - 7s - loss: 1.8318 - accuracy: 0.3809 - val_loss: 1.9464 - val_accurac
y: 0.3356
Epoch 192/200
391/391 - 7s - loss: 1.8347 - accuracy: 0.3788 - val_loss: 1.8085 - val_accurac
y: 0.3900
Epoch 193/200
391/391 - 7s - loss: 1.8350 - accuracy: 0.3797 - val_loss: 1.8849 - val_accurac
y: 0.3520
Epoch 194/200
391/391 - 7s - loss: 1.8378 - accuracy: 0.3779 - val_loss: 1.8096 - val_accurac
y: 0.3841
Epoch 195/200
391/391 - 7s - loss: 1.8309 - accuracy: 0.3807 - val_loss: 1.8177 - val_accurac
y: 0.3727
Epoch 196/200
391/391 - 7s - loss: 1.8322 - accuracy: 0.3788 - val_loss: 1.8402 - val_accurac
y: 0.3718
Epoch 197/200
391/391 - 7s - loss: 1.8360 - accuracy: 0.3801 - val_loss: 1.8381 - val_accurac
```

```
y: 0.3804
Epoch 198/200
391/391 - 7s - loss: 1.8324 - accuracy: 0.3779 - val_loss: 1.8674 - val_accurac
y: 0.3649
Epoch 199/200
391/391 - 7s - loss: 1.8293 - accuracy: 0.3786 - val_loss: 1.8105 - val_accurac
y: 0.3894
Epoch 200/200
391/391 - 7s - loss: 1.8362 - accuracy: 0.3771 - val_loss: 1.9086 - val_accurac
y: 0.3608
```

RMSProp + Nesterov

In [19]:
```
model_nesterov=create_model()
model_nesterov.compile(loss='categorical_crossentropy', optimizer='Nadam', metric
s=['accuracy'])
history_nesterov = model_nesterov.fit(X_train, y_train, batch_size=128, epochs=20
0, validation_data=(X_test,y_test),shuffle=True,verbose=2)
```

```
Epoch 1/200
391/391 - 10s - loss: 8.1082 - accuracy: 0.2711 - val_loss: 2.3732 - val_accurac
y: 0.3345
Epoch 2/200
391/391 - 9s - loss: 2.1051 - accuracy: 0.3321 - val_loss: 1.9524 - val_accurac
y: 0.3469
Epoch 3/200
391/391 - 9s - loss: 1.9206 - accuracy: 0.3541 - val_loss: 1.8379 - val_accurac
y: 0.3778
Epoch 4/200
391/391 - 9s - loss: 1.8759 - accuracy: 0.3666 - val_loss: 2.1163 - val_accurac
y: 0.2945
Epoch 5/200
391/391 - 9s - loss: 1.8498 - accuracy: 0.3746 - val_loss: 1.8967 - val_accurac
y: 0.3539
Epoch 6/200
391/391 - 9s - loss: 1.8384 - accuracy: 0.3770 - val_loss: 1.7833 - val_accurac
y: 0.4021
Epoch 7/200
391/391 - 9s - loss: 1.8186 - accuracy: 0.3846 - val_loss: 1.7874 - val_accurac
y: 0.3977
Epoch 8/200
391/391 - 9s - loss: 1.8050 - accuracy: 0.3917 - val_loss: 1.7620 - val_accurac
y: 0.3998
Epoch 9/200
391/391 - 9s - loss: 1.8036 - accuracy: 0.3934 - val_loss: 1.8428 - val_accurac
y: 0.3715
Epoch 10/200
391/391 - 9s - loss: 1.8239 - accuracy: 0.3906 - val_loss: 1.8223 - val_accurac
y: 0.3862
Epoch 11/200
391/391 - 9s - loss: 1.7945 - accuracy: 0.3982 - val_loss: 1.7455 - val_accurac
y: 0.4168
Epoch 12/200
391/391 - 9s - loss: 1.7765 - accuracy: 0.4022 - val_loss: 1.7619 - val_accurac
y: 0.4149
Epoch 13/200
391/391 - 9s - loss: 1.7607 - accuracy: 0.4089 - val_loss: 2.0450 - val_accurac
y: 0.3066
Epoch 14/200
391/391 - 8s - loss: 1.7550 - accuracy: 0.4093 - val_loss: 1.7309 - val_accurac
y: 0.4144
Epoch 15/200
391/391 - 8s - loss: 1.7351 - accuracy: 0.4154 - val_loss: 1.7281 - val_accurac
y: 0.4229
Epoch 16/200
391/391 - 9s - loss: 1.7244 - accuracy: 0.4184 - val_loss: 1.7092 - val_accurac
y: 0.4307
Epoch 17/200
391/391 - 9s - loss: 1.7195 - accuracy: 0.4212 - val_loss: 1.6873 - val_accurac
y: 0.4386
Epoch 18/200
391/391 - 9s - loss: 1.7252 - accuracy: 0.4234 - val_loss: 1.7686 - val_accurac
y: 0.4158
Epoch 19/200
391/391 - 9s - loss: 1.7308 - accuracy: 0.4192 - val_loss: 1.6873 - val_accurac
y: 0.4445
Epoch 20/200
391/391 - 9s - loss: 1.7096 - accuracy: 0.4258 - val_loss: 1.7055 - val_accurac
```

```
y: 0.4302
Epoch 21/200
391/391 - 9s - loss: 1.7064 - accuracy: 0.4270 - val_loss: 1.7309 - val_accurac
y: 0.4231
Epoch 22/200
391/391 - 9s - loss: 1.6948 - accuracy: 0.4292 - val_loss: 1.7261 - val_accurac
y: 0.4150
Epoch 23/200
391/391 - 9s - loss: 1.6895 - accuracy: 0.4302 - val_loss: 1.7686 - val_accurac
y: 0.4084
Epoch 24/200
391/391 - 9s - loss: 1.6857 - accuracy: 0.4348 - val_loss: 1.7068 - val_accurac
y: 0.4279
Epoch 25/200
391/391 - 9s - loss: 1.6819 - accuracy: 0.4368 - val_loss: 1.6879 - val_accurac
y: 0.4381
Epoch 26/200
391/391 - 9s - loss: 1.6755 - accuracy: 0.4379 - val_loss: 1.6969 - val_accurac
y: 0.4314
Epoch 27/200
391/391 - 9s - loss: 1.6661 - accuracy: 0.4413 - val_loss: 1.6669 - val_accurac
y: 0.4481
Epoch 28/200
391/391 - 9s - loss: 1.6620 - accuracy: 0.4429 - val_loss: 1.7261 - val_accurac
y: 0.4209
Epoch 29/200
391/391 - 9s - loss: 1.6610 - accuracy: 0.4455 - val_loss: 1.6637 - val_accurac
y: 0.4493
Epoch 30/200
391/391 - 9s - loss: 1.6591 - accuracy: 0.4446 - val_loss: 1.6404 - val_accurac
y: 0.4618
Epoch 31/200
391/391 - 9s - loss: 1.6537 - accuracy: 0.4473 - val_loss: 1.7456 - val_accurac
y: 0.3967
Epoch 32/200
391/391 - 9s - loss: 1.6502 - accuracy: 0.4473 - val_loss: 1.6731 - val_accurac
y: 0.4471
Epoch 33/200
391/391 - 9s - loss: 1.6511 - accuracy: 0.4474 - val_loss: 1.7278 - val_accurac
y: 0.4120
Epoch 34/200
391/391 - 9s - loss: 1.6439 - accuracy: 0.4515 - val_loss: 1.7212 - val_accurac
y: 0.4281
Epoch 35/200
391/391 - 9s - loss: 1.6390 - accuracy: 0.4535 - val_loss: 1.6416 - val_accurac
y: 0.4570
Epoch 36/200
391/391 - 9s - loss: 1.6400 - accuracy: 0.4529 - val_loss: 1.6866 - val_accurac
y: 0.4412
Epoch 37/200
391/391 - 9s - loss: 1.6347 - accuracy: 0.4540 - val_loss: 1.6789 - val_accurac
y: 0.4475
Epoch 38/200
391/391 - 9s - loss: 1.6333 - accuracy: 0.4577 - val_loss: 1.6882 - val_accurac
y: 0.4385
Epoch 39/200
391/391 - 9s - loss: 1.6337 - accuracy: 0.4541 - val_loss: 1.6635 - val_accurac
y: 0.4533
Epoch 40/200
```

```
391/391 - 9s - loss: 1.6294 - accuracy: 0.4573 - val_loss: 1.6843 - val_accurac
y: 0.4421
Epoch 41/200
391/391 - 9s - loss: 1.6275 - accuracy: 0.4541 - val_loss: 1.8239 - val_accurac
y: 0.3916
Epoch 42/200
391/391 - 9s - loss: 1.6257 - accuracy: 0.4577 - val_loss: 1.6736 - val_accurac
y: 0.4389
Epoch 43/200
391/391 - 9s - loss: 1.6227 - accuracy: 0.4611 - val_loss: 1.6466 - val_accurac
y: 0.4473
Epoch 44/200
391/391 - 9s - loss: 1.6230 - accuracy: 0.4600 - val_loss: 1.7070 - val_accurac
y: 0.4247
Epoch 45/200
391/391 - 9s - loss: 1.6223 - accuracy: 0.4581 - val_loss: 1.6477 - val_accurac
y: 0.4483
Epoch 46/200
391/391 - 9s - loss: 1.6198 - accuracy: 0.4591 - val_loss: 1.6481 - val_accurac
y: 0.4483
Epoch 47/200
391/391 - 9s - loss: 1.6179 - accuracy: 0.4589 - val_loss: 1.6231 - val_accurac
y: 0.4607
Epoch 48/200
391/391 - 9s - loss: 1.6164 - accuracy: 0.4616 - val_loss: 1.7022 - val_accurac
y: 0.4250
Epoch 49/200
391/391 - 9s - loss: 1.6148 - accuracy: 0.4620 - val_loss: 1.6442 - val_accurac
y: 0.4476
Epoch 50/200
391/391 - 9s - loss: 1.6174 - accuracy: 0.4611 - val_loss: 1.7057 - val_accurac
y: 0.4332
Epoch 51/200
391/391 - 9s - loss: 1.6132 - accuracy: 0.4621 - val_loss: 1.6209 - val_accurac
y: 0.4640
Epoch 52/200
391/391 - 9s - loss: 1.6151 - accuracy: 0.4628 - val_loss: 1.6205 - val_accurac
y: 0.4528
Epoch 53/200
391/391 - 9s - loss: 1.6117 - accuracy: 0.4623 - val_loss: 1.6310 - val_accurac
y: 0.4625
Epoch 54/200
391/391 - 9s - loss: 1.6096 - accuracy: 0.4625 - val_loss: 1.6369 - val_accurac
y: 0.4510
Epoch 55/200
391/391 - 9s - loss: 1.6112 - accuracy: 0.4628 - val_loss: 1.6136 - val_accurac
y: 0.4646
Epoch 56/200
391/391 - 8s - loss: 1.6123 - accuracy: 0.4616 - val_loss: 1.7091 - val_accurac
y: 0.4282
Epoch 57/200
391/391 - 7s - loss: 1.6084 - accuracy: 0.4640 - val_loss: 1.6854 - val_accurac
y: 0.4380
Epoch 58/200
391/391 - 9s - loss: 1.6059 - accuracy: 0.4636 - val_loss: 1.6630 - val_accurac
y: 0.4425
Epoch 59/200
391/391 - 9s - loss: 1.6092 - accuracy: 0.4599 - val_loss: 1.6486 - val_accurac
y: 0.4552
```

```
Epoch 60/200
391/391 - 9s - loss: 1.6072 - accuracy: 0.4634 - val_loss: 1.6137 - val_accurac
y: 0.4655
Epoch 61/200
391/391 - 9s - loss: 1.6042 - accuracy: 0.4640 - val_loss: 1.6670 - val_accurac
y: 0.4375
Epoch 62/200
391/391 - 9s - loss: 1.6043 - accuracy: 0.4653 - val_loss: 1.6555 - val_accurac
y: 0.4414
Epoch 63/200
391/391 - 9s - loss: 1.6031 - accuracy: 0.4625 - val_loss: 1.6573 - val_accurac
y: 0.4373
Epoch 64/200
391/391 - 9s - loss: 1.6034 - accuracy: 0.4653 - val_loss: 1.6565 - val_accurac
y: 0.4466
Epoch 65/200
391/391 - 9s - loss: 1.6020 - accuracy: 0.4652 - val_loss: 1.6072 - val_accurac
y: 0.4710
Epoch 66/200
391/391 - 9s - loss: 1.6012 - accuracy: 0.4653 - val_loss: 1.6402 - val_accurac
y: 0.4545
Epoch 67/200
391/391 - 9s - loss: 1.6025 - accuracy: 0.4650 - val_loss: 1.6644 - val_accurac
y: 0.4374
Epoch 68/200
391/391 - 9s - loss: 1.5981 - accuracy: 0.4689 - val_loss: 1.7220 - val_accurac
y: 0.4353
Epoch 69/200
391/391 - 9s - loss: 1.5989 - accuracy: 0.4650 - val_loss: 1.6336 - val_accurac
y: 0.4590
Epoch 70/200
391/391 - 9s - loss: 1.5994 - accuracy: 0.4627 - val_loss: 1.6363 - val_accurac
y: 0.4602
Epoch 71/200
391/391 - 9s - loss: 1.5979 - accuracy: 0.4666 - val_loss: 1.6277 - val_accurac
y: 0.4590
Epoch 72/200
391/391 - 8s - loss: 1.6016 - accuracy: 0.4637 - val_loss: 1.6236 - val_accurac
y: 0.4605
Epoch 73/200
391/391 - 9s - loss: 1.5956 - accuracy: 0.4657 - val_loss: 1.6248 - val_accurac
y: 0.4603
Epoch 74/200
391/391 - 9s - loss: 1.5974 - accuracy: 0.4655 - val_loss: 1.7062 - val_accurac
y: 0.4324
Epoch 75/200
391/391 - 9s - loss: 1.5994 - accuracy: 0.4647 - val_loss: 1.6389 - val_accurac
y: 0.4473
Epoch 76/200
391/391 - 9s - loss: 1.5979 - accuracy: 0.4665 - val_loss: 1.6254 - val_accurac
y: 0.4611
Epoch 77/200
391/391 - 9s - loss: 1.5945 - accuracy: 0.4686 - val_loss: 1.6077 - val_accurac
y: 0.4638
Epoch 78/200
391/391 - 9s - loss: 1.5964 - accuracy: 0.4670 - val_loss: 1.6116 - val_accurac
y: 0.4619
Epoch 79/200
391/391 - 9s - loss: 1.5924 - accuracy: 0.4693 - val_loss: 1.7286 - val_accurac
```

```
y: 0.4196
Epoch 80/200
391/391 - 9s - loss: 1.5968 - accuracy: 0.4658 - val_loss: 1.6081 - val_accurac
y: 0.4620
Epoch 81/200
391/391 - 9s - loss: 1.5906 - accuracy: 0.4678 - val_loss: 1.5902 - val_accurac
y: 0.4725
Epoch 82/200
391/391 - 9s - loss: 1.5942 - accuracy: 0.4663 - val_loss: 1.6335 - val_accurac
y: 0.4485
Epoch 83/200
391/391 - 9s - loss: 1.5943 - accuracy: 0.4696 - val_loss: 1.6029 - val_accurac
y: 0.4646
Epoch 84/200
391/391 - 9s - loss: 1.5913 - accuracy: 0.4668 - val_loss: 1.6142 - val_accurac
y: 0.4655
Epoch 85/200
391/391 - 9s - loss: 1.5936 - accuracy: 0.4673 - val_loss: 1.6126 - val_accurac
y: 0.4657
Epoch 86/200
391/391 - 9s - loss: 1.5945 - accuracy: 0.4683 - val_loss: 1.6185 - val_accurac
y: 0.4580
Epoch 87/200
391/391 - 9s - loss: 1.5907 - accuracy: 0.4687 - val_loss: 1.6276 - val_accurac
y: 0.4573
Epoch 88/200
391/391 - 9s - loss: 1.5918 - accuracy: 0.4673 - val_loss: 1.6136 - val_accurac
y: 0.4620
Epoch 89/200
391/391 - 9s - loss: 1.5910 - accuracy: 0.4677 - val_loss: 1.6103 - val_accurac
y: 0.4629
Epoch 90/200
391/391 - 9s - loss: 1.5886 - accuracy: 0.4687 - val_loss: 1.6043 - val_accurac
y: 0.4646
Epoch 91/200
391/391 - 9s - loss: 1.5908 - accuracy: 0.4673 - val_loss: 1.6573 - val_accurac
y: 0.4479
Epoch 92/200
391/391 - 9s - loss: 1.5885 - accuracy: 0.4688 - val_loss: 1.5989 - val_accurac
y: 0.4694
Epoch 93/200
391/391 - 9s - loss: 1.5935 - accuracy: 0.4684 - val_loss: 1.6046 - val_accurac
y: 0.4700
Epoch 94/200
391/391 - 9s - loss: 1.5887 - accuracy: 0.4693 - val_loss: 1.6738 - val_accurac
y: 0.4333
Epoch 95/200
391/391 - 9s - loss: 1.5873 - accuracy: 0.4677 - val_loss: 1.6271 - val_accurac
y: 0.4466
Epoch 96/200
391/391 - 9s - loss: 1.5883 - accuracy: 0.4678 - val_loss: 1.6495 - val_accurac
y: 0.4493
Epoch 97/200
391/391 - 9s - loss: 1.5867 - accuracy: 0.4696 - val_loss: 1.5918 - val_accurac
y: 0.4684
Epoch 98/200
391/391 - 9s - loss: 1.5859 - accuracy: 0.4718 - val_loss: 1.6081 - val_accurac
y: 0.4684
Epoch 99/200
```

```
391/391 - 8s - loss: 1.5871 - accuracy: 0.4708 - val_loss: 1.5937 - val_accurac
y: 0.4672
Epoch 100/200
391/391 - 8s - loss: 1.5870 - accuracy: 0.4717 - val_loss: 1.6208 - val_accurac
y: 0.4610
Epoch 101/200
391/391 - 9s - loss: 1.5839 - accuracy: 0.4716 - val_loss: 1.6439 - val_accurac
y: 0.4532
Epoch 102/200
391/391 - 9s - loss: 1.5874 - accuracy: 0.4660 - val_loss: 1.5888 - val_accurac
y: 0.4699
Epoch 103/200
391/391 - 9s - loss: 1.5864 - accuracy: 0.4695 - val_loss: 1.5934 - val_accurac
y: 0.4711
Epoch 104/200
391/391 - 9s - loss: 1.5859 - accuracy: 0.4711 - val_loss: 1.7186 - val_accurac
y: 0.4301
Epoch 105/200
391/391 - 9s - loss: 1.5851 - accuracy: 0.4681 - val_loss: 1.6203 - val_accurac
y: 0.4559
Epoch 106/200
391/391 - 9s - loss: 1.5861 - accuracy: 0.4703 - val_loss: 1.6004 - val_accurac
y: 0.4661
Epoch 107/200
391/391 - 9s - loss: 1.5842 - accuracy: 0.4705 - val_loss: 1.6198 - val_accurac
y: 0.4538
Epoch 108/200
391/391 - 9s - loss: 1.5812 - accuracy: 0.4721 - val_loss: 1.6196 - val_accurac
y: 0.4571
Epoch 109/200
391/391 - 9s - loss: 1.5859 - accuracy: 0.4711 - val_loss: 1.6277 - val_accurac
y: 0.4578
Epoch 110/200
391/391 - 9s - loss: 1.5866 - accuracy: 0.4680 - val_loss: 1.6009 - val_accurac
y: 0.4618
Epoch 111/200
391/391 - 9s - loss: 1.5826 - accuracy: 0.4703 - val_loss: 1.6162 - val_accurac
y: 0.4559
Epoch 112/200
391/391 - 9s - loss: 1.5847 - accuracy: 0.4699 - val_loss: 1.5872 - val_accurac
y: 0.4773
Epoch 113/200
391/391 - 9s - loss: 1.5832 - accuracy: 0.4722 - val_loss: 1.6334 - val_accurac
y: 0.4568
Epoch 114/200
391/391 - 9s - loss: 1.5821 - accuracy: 0.4729 - val_loss: 1.6197 - val_accurac
y: 0.4560
Epoch 115/200
391/391 - 9s - loss: 1.5832 - accuracy: 0.4694 - val_loss: 1.6204 - val_accurac
y: 0.4562
Epoch 116/200
391/391 - 9s - loss: 1.5845 - accuracy: 0.4683 - val_loss: 1.6414 - val_accurac
y: 0.4442
Epoch 117/200
391/391 - 9s - loss: 1.5846 - accuracy: 0.4702 - val_loss: 1.6799 - val_accurac
y: 0.4432
Epoch 118/200
391/391 - 9s - loss: 1.5831 - accuracy: 0.4682 - val_loss: 1.6177 - val_accurac
y: 0.4606
```

```
Epoch 119/200
391/391 - 9s - loss: 1.5816 - accuracy: 0.4726 - val_loss: 1.5788 - val_accurac
y: 0.4762
Epoch 120/200
391/391 - 9s - loss: 1.5835 - accuracy: 0.4721 - val_loss: 1.6309 - val_accurac
y: 0.4572
Epoch 121/200
391/391 - 9s - loss: 1.5803 - accuracy: 0.4712 - val_loss: 1.6215 - val_accurac
y: 0.4585
Epoch 122/200
391/391 - 9s - loss: 1.5779 - accuracy: 0.4712 - val_loss: 1.6304 - val_accurac
y: 0.4535
Epoch 123/200
391/391 - 9s - loss: 1.5815 - accuracy: 0.4722 - val_loss: 1.6173 - val_accurac
y: 0.4612
Epoch 124/200
391/391 - 9s - loss: 1.5782 - accuracy: 0.4740 - val_loss: 1.6119 - val_accurac
y: 0.4528
Epoch 125/200
391/391 - 9s - loss: 1.5796 - accuracy: 0.4711 - val_loss: 1.6328 - val_accurac
y: 0.4488
Epoch 126/200
391/391 - 9s - loss: 1.5801 - accuracy: 0.4705 - val_loss: 1.6069 - val_accurac
y: 0.4620
Epoch 127/200
391/391 - 9s - loss: 1.5772 - accuracy: 0.4722 - val_loss: 1.6691 - val_accurac
y: 0.4381
Epoch 128/200
391/391 - 9s - loss: 1.5814 - accuracy: 0.4703 - val_loss: 1.6353 - val_accurac
y: 0.4557
Epoch 129/200
391/391 - 9s - loss: 1.5801 - accuracy: 0.4711 - val_loss: 1.6235 - val_accurac
y: 0.4599
Epoch 130/200
391/391 - 9s - loss: 1.5801 - accuracy: 0.4714 - val_loss: 1.5915 - val_accurac
y: 0.4701
Epoch 131/200
391/391 - 9s - loss: 1.5813 - accuracy: 0.4722 - val_loss: 1.6282 - val_accurac
y: 0.4547
Epoch 132/200
391/391 - 9s - loss: 1.5767 - accuracy: 0.4731 - val_loss: 1.6423 - val_accurac
y: 0.4494
Epoch 133/200
391/391 - 9s - loss: 1.5777 - accuracy: 0.4710 - val_loss: 1.6412 - val_accurac
y: 0.4568
Epoch 134/200
391/391 - 9s - loss: 1.5780 - accuracy: 0.4735 - val_loss: 1.6897 - val_accurac
y: 0.4351
Epoch 135/200
391/391 - 9s - loss: 1.5818 - accuracy: 0.4717 - val_loss: 1.6039 - val_accurac
y: 0.4687
Epoch 136/200
391/391 - 9s - loss: 1.5779 - accuracy: 0.4722 - val_loss: 1.6595 - val_accurac
y: 0.4433
Epoch 137/200
391/391 - 9s - loss: 1.5764 - accuracy: 0.4711 - val_loss: 1.6227 - val_accurac
y: 0.4591
Epoch 138/200
391/391 - 9s - loss: 1.5764 - accuracy: 0.4750 - val_loss: 1.6286 - val_accurac
```

```
y: 0.4590
Epoch 139/200
391/391 - 9s - loss: 1.5794 - accuracy: 0.4733 - val_loss: 1.5793 - val_accurac
y: 0.4792
Epoch 140/200
391/391 - 9s - loss: 1.5762 - accuracy: 0.4723 - val_loss: 1.6197 - val_accurac
y: 0.4555
Epoch 141/200
391/391 - 8s - loss: 1.5753 - accuracy: 0.4747 - val_loss: 1.6005 - val_accurac
y: 0.4717
Epoch 142/200
391/391 - 8s - loss: 1.5767 - accuracy: 0.4741 - val_loss: 1.6290 - val_accurac
y: 0.4531
Epoch 143/200
391/391 - 9s - loss: 1.5756 - accuracy: 0.4752 - val_loss: 1.6448 - val_accurac
y: 0.4503
Epoch 144/200
391/391 - 9s - loss: 1.5754 - accuracy: 0.4730 - val_loss: 1.6117 - val_accurac
y: 0.4605
Epoch 145/200
391/391 - 9s - loss: 1.5769 - accuracy: 0.4730 - val_loss: 1.6167 - val_accurac
y: 0.4582
Epoch 146/200
391/391 - 9s - loss: 1.5771 - accuracy: 0.4711 - val_loss: 1.6276 - val_accurac
y: 0.4564
Epoch 147/200
391/391 - 9s - loss: 1.5727 - accuracy: 0.4756 - val_loss: 1.6245 - val_accurac
y: 0.4540
Epoch 148/200
391/391 - 9s - loss: 1.5720 - accuracy: 0.4755 - val_loss: 1.6009 - val_accurac
y: 0.4680
Epoch 149/200
391/391 - 9s - loss: 1.5772 - accuracy: 0.4715 - val_loss: 1.5786 - val_accurac
y: 0.4711
Epoch 150/200
391/391 - 10s - loss: 1.5747 - accuracy: 0.4720 - val_loss: 1.5781 - val_accurac
y: 0.4760
Epoch 151/200
391/391 - 9s - loss: 1.5751 - accuracy: 0.4727 - val_loss: 1.5909 - val_accurac
y: 0.4724
Epoch 152/200
391/391 - 9s - loss: 1.5749 - accuracy: 0.4744 - val_loss: 1.6568 - val_accurac
y: 0.4403
Epoch 153/200
391/391 - 9s - loss: 1.5762 - accuracy: 0.4718 - val_loss: 1.5939 - val_accurac
y: 0.4675
Epoch 154/200
391/391 - 9s - loss: 1.5773 - accuracy: 0.4720 - val_loss: 1.7241 - val_accurac
y: 0.4291
Epoch 155/200
391/391 - 9s - loss: 1.5728 - accuracy: 0.4740 - val_loss: 1.6130 - val_accurac
y: 0.4586
Epoch 156/200
391/391 - 9s - loss: 1.5746 - accuracy: 0.4733 - val_loss: 1.5993 - val_accurac
y: 0.4673
Epoch 157/200
391/391 - 9s - loss: 1.5741 - accuracy: 0.4754 - val_loss: 1.6199 - val_accurac
y: 0.4577
Epoch 158/200
```

```
391/391 - 9s - loss: 1.5749 - accuracy: 0.4747 - val_loss: 1.6705 - val_accurac
y: 0.4350
Epoch 159/200
391/391 - 9s - loss: 1.5733 - accuracy: 0.4738 - val_loss: 1.6150 - val_accurac
y: 0.4648
Epoch 160/200
391/391 - 9s - loss: 1.5735 - accuracy: 0.4731 - val_loss: 1.6903 - val_accurac
y: 0.4346
Epoch 161/200
391/391 - 9s - loss: 1.5715 - accuracy: 0.4741 - val_loss: 1.5960 - val_accurac
y: 0.4677
Epoch 162/200
391/391 - 9s - loss: 1.5732 - accuracy: 0.4739 - val_loss: 1.5741 - val_accurac
y: 0.4797
Epoch 163/200
391/391 - 9s - loss: 1.5708 - accuracy: 0.4756 - val_loss: 1.6401 - val_accurac
y: 0.4513
Epoch 164/200
391/391 - 9s - loss: 1.5780 - accuracy: 0.4728 - val_loss: 1.6078 - val_accurac
y: 0.4668
Epoch 165/200
391/391 - 9s - loss: 1.5716 - accuracy: 0.4760 - val_loss: 1.6106 - val_accurac
y: 0.4609
Epoch 166/200
391/391 - 9s - loss: 1.5703 - accuracy: 0.4771 - val_loss: 1.6342 - val_accurac
y: 0.4584
Epoch 167/200
391/391 - 9s - loss: 1.5721 - accuracy: 0.4735 - val_loss: 1.6005 - val_accurac
y: 0.4655
Epoch 168/200
391/391 - 9s - loss: 1.5713 - accuracy: 0.4751 - val_loss: 1.6231 - val_accurac
y: 0.4568
Epoch 169/200
391/391 - 9s - loss: 1.5733 - accuracy: 0.4734 - val_loss: 1.6158 - val_accurac
y: 0.4615
Epoch 170/200
391/391 - 9s - loss: 1.5742 - accuracy: 0.4733 - val_loss: 1.6118 - val_accurac
y: 0.4602
Epoch 171/200
391/391 - 9s - loss: 1.5718 - accuracy: 0.4746 - val_loss: 1.6010 - val_accurac
y: 0.4666
Epoch 172/200
391/391 - 9s - loss: 1.5714 - accuracy: 0.4756 - val_loss: 1.5999 - val_accurac
y: 0.4664
Epoch 173/200
391/391 - 9s - loss: 1.5737 - accuracy: 0.4756 - val_loss: 1.6040 - val_accurac
y: 0.4677
Epoch 174/200
391/391 - 9s - loss: 1.5695 - accuracy: 0.4751 - val_loss: 1.6039 - val_accurac
y: 0.4649
Epoch 175/200
391/391 - 9s - loss: 1.5710 - accuracy: 0.4739 - val_loss: 1.6501 - val_accurac
y: 0.4497
Epoch 176/200
391/391 - 9s - loss: 1.5703 - accuracy: 0.4738 - val_loss: 1.6441 - val_accurac
y: 0.4521
Epoch 177/200
391/391 - 9s - loss: 1.5726 - accuracy: 0.4740 - val_loss: 1.6681 - val_accurac
y: 0.4370
```

```
Epoch 178/200
391/391 - 9s - loss: 1.5718 - accuracy: 0.4762 - val_loss: 1.6281 - val_accurac
y: 0.4496
Epoch 179/200
391/391 - 9s - loss: 1.5715 - accuracy: 0.4749 - val_loss: 1.5873 - val_accurac
y: 0.4646
Epoch 180/200
391/391 - 9s - loss: 1.5683 - accuracy: 0.4767 - val_loss: 1.6137 - val_accurac
y: 0.4566
Epoch 181/200
391/391 - 9s - loss: 1.5740 - accuracy: 0.4731 - val_loss: 1.6420 - val_accurac
y: 0.4540
Epoch 182/200
391/391 - 9s - loss: 1.5726 - accuracy: 0.4726 - val_loss: 1.6458 - val_accurac
y: 0.4493
Epoch 183/200
391/391 - 8s - loss: 1.5684 - accuracy: 0.4743 - val_loss: 1.5902 - val_accurac
y: 0.4651
Epoch 184/200
391/391 - 8s - loss: 1.5722 - accuracy: 0.4725 - val_loss: 1.6491 - val_accurac
y: 0.4476
Epoch 185/200
391/391 - 8s - loss: 1.5709 - accuracy: 0.4740 - val_loss: 1.6113 - val_accurac
y: 0.4583
Epoch 186/200
391/391 - 9s - loss: 1.5690 - accuracy: 0.4763 - val_loss: 1.5744 - val_accurac
y: 0.4827
Epoch 187/200
391/391 - 9s - loss: 1.5689 - accuracy: 0.4728 - val_loss: 1.6276 - val_accurac
y: 0.4581
Epoch 188/200
391/391 - 9s - loss: 1.5704 - accuracy: 0.4758 - val_loss: 1.6541 - val_accurac
y: 0.4548
Epoch 189/200
391/391 - 9s - loss: 1.5673 - accuracy: 0.4761 - val_loss: 1.5777 - val_accurac
y: 0.4774
Epoch 190/200
391/391 - 9s - loss: 1.5684 - accuracy: 0.4739 - val_loss: 1.6471 - val_accurac
y: 0.4554
Epoch 191/200
391/391 - 9s - loss: 1.5733 - accuracy: 0.4751 - val_loss: 1.6733 - val_accurac
y: 0.4456
Epoch 192/200
391/391 - 9s - loss: 1.5681 - accuracy: 0.4753 - val_loss: 1.6180 - val_accurac
y: 0.4629
Epoch 193/200
391/391 - 9s - loss: 1.5678 - accuracy: 0.4755 - val_loss: 1.5749 - val_accurac
y: 0.4776
Epoch 194/200
391/391 - 9s - loss: 1.5691 - accuracy: 0.4754 - val_loss: 1.5898 - val_accurac
y: 0.4707
Epoch 195/200
391/391 - 9s - loss: 1.5711 - accuracy: 0.4739 - val_loss: 1.5846 - val_accurac
y: 0.4714
Epoch 196/200
391/391 - 9s - loss: 1.5690 - accuracy: 0.4753 - val_loss: 1.6264 - val_accurac
y: 0.4553
Epoch 197/200
391/391 - 9s - loss: 1.5718 - accuracy: 0.4745 - val_loss: 1.6159 - val_accurac
```

```
y: 0.4645
Epoch 198/200
391/391 - 9s - loss: 1.5715 - accuracy: 0.4743 - val_loss: 1.5840 - val_accurac
y: 0.4733
Epoch 199/200
391/391 - 9s - loss: 1.5653 - accuracy: 0.4764 - val_loss: 1.6632 - val_accurac
y: 0.4398
Epoch 200/200
391/391 - 9s - loss: 1.5673 - accuracy: 0.4761 - val_loss: 1.5688 - val_accurac
y: 0.4755
```

Adadelta

In [20]:

```
model_adadelta=create_model()
model_adadelta.compile(loss='categorical_crossentropy', optimizer='Adadelta', metrics=['accuracy'])
history_adadelta = model_adadelta.fit(X_train, y_train, batch_size=128, epochs=200, validation_data=(X_test,y_test),shuffle=True,verbose=2)
```

```
Epoch 1/200
391/391 - 7s - loss: 42.0323 - accuracy: 0.1840 - val_loss: 41.7393 - val_accura
cy: 0.2329
Epoch 2/200
391/391 - 6s - loss: 41.4770 - accuracy: 0.2553 - val_loss: 41.2195 - val_accura
cy: 0.2799
Epoch 3/200
391/391 - 6s - loss: 40.9734 - accuracy: 0.2870 - val_loss: 40.7289 - val_accura
cy: 0.2999
Epoch 4/200
391/391 - 6s - loss: 40.4943 - accuracy: 0.3073 - val_loss: 40.2596 - val_accura
cy: 0.3183
Epoch 5/200
391/391 - 6s - loss: 40.0313 - accuracy: 0.3228 - val_loss: 39.8029 - val_accura
cy: 0.3306
Epoch 6/200
391/391 - 5s - loss: 39.5803 - accuracy: 0.3326 - val_loss: 39.3582 - val_accura
cy: 0.3403
Epoch 7/200
391/391 - 6s - loss: 39.1396 - accuracy: 0.3403 - val_loss: 38.9218 - val_accura
cy: 0.3466
Epoch 8/200
391/391 - 6s - loss: 38.7072 - accuracy: 0.3463 - val_loss: 38.4937 - val_accura
cy: 0.3515
Epoch 9/200
391/391 - 6s - loss: 38.2828 - accuracy: 0.3528 - val_loss: 38.0736 - val_accura
cy: 0.3576
Epoch 10/200
391/391 - 6s - loss: 37.8664 - accuracy: 0.3578 - val_loss: 37.6601 - val_accura
cy: 0.3629
Epoch 11/200
391/391 - 6s - loss: 37.4556 - accuracy: 0.3609 - val_loss: 37.2537 - val_accura
cy: 0.3674
Epoch 12/200
391/391 - 6s - loss: 37.0523 - accuracy: 0.3636 - val_loss: 36.8538 - val_accura
cy: 0.3699
Epoch 13/200
391/391 - 6s - loss: 36.6560 - accuracy: 0.3675 - val_loss: 36.4614 - val_accura
cy: 0.3740
Epoch 14/200
391/391 - 6s - loss: 36.2656 - accuracy: 0.3705 - val_loss: 36.0735 - val_accura
cy: 0.3735
Epoch 15/200
391/391 - 6s - loss: 35.8807 - accuracy: 0.3733 - val_loss: 35.6920 - val_accura
cy: 0.3773
Epoch 16/200
391/391 - 6s - loss: 35.5019 - accuracy: 0.3768 - val_loss: 35.3157 - val_accura
cy: 0.3798
Epoch 17/200
391/391 - 6s - loss: 35.1287 - accuracy: 0.3786 - val_loss: 34.9459 - val_accura
cy: 0.3842
Epoch 18/200
391/391 - 6s - loss: 34.7606 - accuracy: 0.3810 - val_loss: 34.5805 - val_accura
cy: 0.3850
Epoch 19/200
391/391 - 6s - loss: 34.3969 - accuracy: 0.3832 - val_loss: 34.2184 - val_accura
cy: 0.3859
Epoch 20/200
391/391 - 6s - loss: 34.0383 - accuracy: 0.3850 - val_loss: 33.8636 - val_accura
```

```
cy: 0.3869
Epoch 21/200
391/391 - 6s - loss: 33.6855 - accuracy: 0.3870 - val_loss: 33.5129 - val_accura
cy: 0.3900
Epoch 22/200
391/391 - 6s - loss: 33.3375 - accuracy: 0.3888 - val_loss: 33.1679 - val_accura
cy: 0.3934
Epoch 23/200
391/391 - 6s - loss: 32.9932 - accuracy: 0.3909 - val_loss: 32.8254 - val_accura
cy: 0.3903
Epoch 24/200
391/391 - 6s - loss: 32.6535 - accuracy: 0.3921 - val_loss: 32.4881 - val_accura
cy: 0.3916
Epoch 25/200
391/391 - 6s - loss: 32.3179 - accuracy: 0.3932 - val_loss: 32.1544 - val_accura
cy: 0.3941
Epoch 26/200
391/391 - 6s - loss: 31.9873 - accuracy: 0.3954 - val_loss: 31.8271 - val_accura
cy: 0.3970
Epoch 27/200
391/391 - 6s - loss: 31.6619 - accuracy: 0.3964 - val_loss: 31.5043 - val_accura
cy: 0.3998
Epoch 28/200
391/391 - 6s - loss: 31.3415 - accuracy: 0.3989 - val_loss: 31.1854 - val_accura
cy: 0.3983
Epoch 29/200
391/391 - 6s - loss: 31.0239 - accuracy: 0.3988 - val_loss: 30.8693 - val_accura
cy: 0.4004
Epoch 30/200
391/391 - 6s - loss: 30.7099 - accuracy: 0.4005 - val_loss: 30.5585 - val_accura
cy: 0.4011
Epoch 31/200
391/391 - 6s - loss: 30.4004 - accuracy: 0.4007 - val_loss: 30.2509 - val_accura
cy: 0.4023
Epoch 32/200
391/391 - 6s - loss: 30.0942 - accuracy: 0.4025 - val_loss: 29.9462 - val_accura
cy: 0.4043
Epoch 33/200
391/391 - 6s - loss: 29.7914 - accuracy: 0.4046 - val_loss: 29.6455 - val_accura
cy: 0.4051
Epoch 34/200
391/391 - 6s - loss: 29.4923 - accuracy: 0.4046 - val_loss: 29.3482 - val_accura
cy: 0.4066
Epoch 35/200
391/391 - 6s - loss: 29.1976 - accuracy: 0.4057 - val_loss: 29.0560 - val_accura
cy: 0.4077
Epoch 36/200
391/391 - 6s - loss: 28.9062 - accuracy: 0.4064 - val_loss: 28.7655 - val_accura
cy: 0.4089
Epoch 37/200
391/391 - 6s - loss: 28.6178 - accuracy: 0.4071 - val_loss: 28.4794 - val_accura
cy: 0.4068
Epoch 38/200
391/391 - 5s - loss: 28.3335 - accuracy: 0.4092 - val_loss: 28.1977 - val_accura
cy: 0.4111
Epoch 39/200
391/391 - 5s - loss: 28.0531 - accuracy: 0.4095 - val_loss: 27.9192 - val_accura
cy: 0.4121
Epoch 40/200
```

```
391/391 - 6s - loss: 27.7762 - accuracy: 0.4104 - val_loss: 27.6440 - val_accura
cy: 0.4098
Epoch 41/200
391/391 - 6s - loss: 27.5021 - accuracy: 0.4102 - val_loss: 27.3718 - val_accura
cy: 0.4120
Epoch 42/200
391/391 - 6s - loss: 27.2316 - accuracy: 0.4116 - val_loss: 27.1026 - val_accura
cy: 0.4092
Epoch 43/200
391/391 - 6s - loss: 26.9645 - accuracy: 0.4127 - val_loss: 26.8381 - val_accura
cy: 0.4115
Epoch 44/200
391/391 - 6s - loss: 26.7006 - accuracy: 0.4150 - val_loss: 26.5750 - val_accura
cy: 0.4134
Epoch 45/200
391/391 - 6s - loss: 26.4391 - accuracy: 0.4134 - val_loss: 26.3153 - val_accura
cy: 0.4138
Epoch 46/200
391/391 - 6s - loss: 26.1813 - accuracy: 0.4152 - val_loss: 26.0584 - val_accura
cy: 0.4154
Epoch 47/200
391/391 - 6s - loss: 25.9257 - accuracy: 0.4158 - val_loss: 25.8035 - val_accura
cy: 0.4165
Epoch 48/200
391/391 - 6s - loss: 25.6730 - accuracy: 0.4172 - val_loss: 25.5537 - val_accura
cy: 0.4148
Epoch 49/200
391/391 - 6s - loss: 25.4244 - accuracy: 0.4169 - val_loss: 25.3065 - val_accura
cy: 0.4169
Epoch 50/200
391/391 - 6s - loss: 25.1787 - accuracy: 0.4169 - val_loss: 25.0624 - val_accura
cy: 0.4198
Epoch 51/200
391/391 - 6s - loss: 24.9358 - accuracy: 0.4187 - val_loss: 24.8207 - val_accura
cy: 0.4190
Epoch 52/200
391/391 - 6s - loss: 24.6951 - accuracy: 0.4191 - val_loss: 24.5821 - val_accura
cy: 0.4186
Epoch 53/200
391/391 - 6s - loss: 24.4575 - accuracy: 0.4195 - val_loss: 24.3459 - val_accura
cy: 0.4175
Epoch 54/200
391/391 - 6s - loss: 24.2226 - accuracy: 0.4206 - val_loss: 24.1123 - val_accura
cy: 0.4204
Epoch 55/200
391/391 - 6s - loss: 23.9910 - accuracy: 0.4210 - val_loss: 23.8817 - val_accura
cy: 0.4202
Epoch 56/200
391/391 - 6s - loss: 23.7616 - accuracy: 0.4218 - val_loss: 23.6543 - val_accura
cy: 0.4193
Epoch 57/200
391/391 - 6s - loss: 23.5348 - accuracy: 0.4217 - val_loss: 23.4302 - val_accura
cy: 0.4199
Epoch 58/200
391/391 - 6s - loss: 23.3106 - accuracy: 0.4223 - val_loss: 23.2055 - val_accura
cy: 0.4234
Epoch 59/200
391/391 - 6s - loss: 23.0883 - accuracy: 0.4228 - val_loss: 22.9845 - val_accura
cy: 0.4225
```

```
Epoch 60/200
391/391 - 6s - loss: 22.8683 - accuracy: 0.4243 - val_loss: 22.7671 - val_accura
cy: 0.4213
Epoch 61/200
391/391 - 6s - loss: 22.6516 - accuracy: 0.4248 - val_loss: 22.5509 - val_accura
cy: 0.4223
Epoch 62/200
391/391 - 6s - loss: 22.4372 - accuracy: 0.4259 - val_loss: 22.3379 - val_accura
cy: 0.4235
Epoch 63/200
391/391 - 6s - loss: 22.2255 - accuracy: 0.4250 - val_loss: 22.1278 - val_accura
cy: 0.4223
Epoch 64/200
391/391 - 6s - loss: 22.0156 - accuracy: 0.4261 - val_loss: 21.9183 - val_accura
cy: 0.4229
Epoch 65/200
391/391 - 6s - loss: 21.8078 - accuracy: 0.4264 - val_loss: 21.7121 - val_accura
cy: 0.4237
Epoch 66/200
391/391 - 6s - loss: 21.6030 - accuracy: 0.4269 - val_loss: 21.5082 - val_accura
cy: 0.4261
Epoch 67/200
391/391 - 6s - loss: 21.4006 - accuracy: 0.4280 - val_loss: 21.3074 - val_accura
cy: 0.4259
Epoch 68/200
391/391 - 6s - loss: 21.2002 - accuracy: 0.4269 - val_loss: 21.1085 - val_accura
cy: 0.4264
Epoch 69/200
391/391 - 6s - loss: 21.0024 - accuracy: 0.4279 - val_loss: 20.9120 - val_accura
cy: 0.4251
Epoch 70/200
391/391 - 6s - loss: 20.8063 - accuracy: 0.4284 - val_loss: 20.7171 - val_accura
cy: 0.4267
Epoch 71/200
391/391 - 6s - loss: 20.6125 - accuracy: 0.4291 - val_loss: 20.5243 - val_accura
cy: 0.4276
Epoch 72/200
391/391 - 6s - loss: 20.4206 - accuracy: 0.4294 - val_loss: 20.3336 - val_accura
cy: 0.4278
Epoch 73/200
391/391 - 6s - loss: 20.2310 - accuracy: 0.4308 - val_loss: 20.1458 - val_accura
cy: 0.4253
Epoch 74/200
391/391 - 6s - loss: 20.0434 - accuracy: 0.4302 - val_loss: 19.9583 - val_accura
cy: 0.4265
Epoch 75/200
391/391 - 6s - loss: 19.8576 - accuracy: 0.4305 - val_loss: 19.7736 - val_accura
cy: 0.4295
Epoch 76/200
391/391 - 6s - loss: 19.6742 - accuracy: 0.4313 - val_loss: 19.5918 - val_accura
cy: 0.4287
Epoch 77/200
391/391 - 6s - loss: 19.4935 - accuracy: 0.4310 - val_loss: 19.4124 - val_accura
cy: 0.4253
Epoch 78/200
391/391 - 6s - loss: 19.3147 - accuracy: 0.4322 - val_loss: 19.2348 - val_accura
cy: 0.4295
Epoch 79/200
391/391 - 6s - loss: 19.1376 - accuracy: 0.4327 - val_loss: 19.0585 - val_accura
```

```
cy: 0.4279
Epoch 80/200
391/391 - 6s - loss: 18.9625 - accuracy: 0.4340 - val_loss: 18.8844 - val_accura
cy: 0.4284
Epoch 81/200
391/391 - 6s - loss: 18.7896 - accuracy: 0.4329 - val_loss: 18.7132 - val_accura
cy: 0.4306
Epoch 82/200
391/391 - 6s - loss: 18.6184 - accuracy: 0.4332 - val_loss: 18.5416 - val_accura
cy: 0.4284
Epoch 83/200
391/391 - 6s - loss: 18.4483 - accuracy: 0.4343 - val_loss: 18.3731 - val_accura
cy: 0.4299
Epoch 84/200
391/391 - 6s - loss: 18.2804 - accuracy: 0.4344 - val_loss: 18.2062 - val_accura
cy: 0.4297
Epoch 85/200
391/391 - 6s - loss: 18.1144 - accuracy: 0.4348 - val_loss: 18.0410 - val_accura
cy: 0.4300
Epoch 86/200
391/391 - 6s - loss: 17.9505 - accuracy: 0.4357 - val_loss: 17.8786 - val_accura
cy: 0.4319
Epoch 87/200
391/391 - 6s - loss: 17.7886 - accuracy: 0.4353 - val_loss: 17.7172 - val_accura
cy: 0.4324
Epoch 88/200
391/391 - 6s - loss: 17.6283 - accuracy: 0.4359 - val_loss: 17.5575 - val_accura
cy: 0.4301
Epoch 89/200
391/391 - 6s - loss: 17.4690 - accuracy: 0.4366 - val_loss: 17.3988 - val_accura
cy: 0.4318
Epoch 90/200
391/391 - 6s - loss: 17.3116 - accuracy: 0.4368 - val_loss: 17.2430 - val_accura
cy: 0.4324
Epoch 91/200
391/391 - 6s - loss: 17.1563 - accuracy: 0.4376 - val_loss: 17.0885 - val_accura
cy: 0.4346
Epoch 92/200
391/391 - 6s - loss: 17.0026 - accuracy: 0.4389 - val_loss: 16.9360 - val_accura
cy: 0.4343
Epoch 93/200
391/391 - 5s - loss: 16.8509 - accuracy: 0.4380 - val_loss: 16.7857 - val_accura
cy: 0.4354
Epoch 94/200
391/391 - 6s - loss: 16.7007 - accuracy: 0.4391 - val_loss: 16.6366 - val_accura
cy: 0.4339
Epoch 95/200
391/391 - 6s - loss: 16.5516 - accuracy: 0.4396 - val_loss: 16.4879 - val_accura
cy: 0.4338
Epoch 96/200
391/391 - 5s - loss: 16.4039 - accuracy: 0.4399 - val_loss: 16.3415 - val_accura
cy: 0.4320
Epoch 97/200
391/391 - 6s - loss: 16.2584 - accuracy: 0.4395 - val_loss: 16.1966 - val_accura
cy: 0.4343
Epoch 98/200
391/391 - 6s - loss: 16.1140 - accuracy: 0.4400 - val_loss: 16.0527 - val_accura
cy: 0.4345
Epoch 99/200
```

```
391/391 - 6s - loss: 15.9710 - accuracy: 0.4402 - val_loss: 15.9104 - val_accura
cy: 0.4326
Epoch 100/200
391/391 - 5s - loss: 15.8298 - accuracy: 0.4403 - val_loss: 15.7710 - val_accura
cy: 0.4329
Epoch 101/200
391/391 - 5s - loss: 15.6908 - accuracy: 0.4408 - val_loss: 15.6318 - val_accura
cy: 0.4341
Epoch 102/200
391/391 - 5s - loss: 15.5527 - accuracy: 0.4410 - val_loss: 15.4944 - val_accura
cy: 0.4329
Epoch 103/200
391/391 - 6s - loss: 15.4162 - accuracy: 0.4414 - val_loss: 15.3590 - val_accura
cy: 0.4339
Epoch 104/200
391/391 - 5s - loss: 15.2809 - accuracy: 0.4413 - val_loss: 15.2256 - val_accura
cy: 0.4359
Epoch 105/200
391/391 - 6s - loss: 15.1472 - accuracy: 0.4415 - val_loss: 15.0910 - val_accura
cy: 0.4357
Epoch 106/200
391/391 - 6s - loss: 15.0143 - accuracy: 0.4425 - val_loss: 14.9594 - val_accura
cy: 0.4358
Epoch 107/200
391/391 - 6s - loss: 14.8832 - accuracy: 0.4410 - val_loss: 14.8289 - val_accura
cy: 0.4372
Epoch 108/200
391/391 - 6s - loss: 14.7541 - accuracy: 0.4424 - val_loss: 14.7013 - val_accura
cy: 0.4357
Epoch 109/200
391/391 - 6s - loss: 14.6258 - accuracy: 0.4428 - val_loss: 14.5729 - val_accura
cy: 0.4362
Epoch 110/200
391/391 - 6s - loss: 14.4991 - accuracy: 0.4438 - val_loss: 14.4485 - val_accura
cy: 0.4350
Epoch 111/200
391/391 - 6s - loss: 14.3742 - accuracy: 0.4437 - val_loss: 14.3225 - val_accura
cy: 0.4382
Epoch 112/200
391/391 - 6s - loss: 14.2500 - accuracy: 0.4442 - val_loss: 14.2004 - val_accura
cy: 0.4378
Epoch 113/200
391/391 - 6s - loss: 14.1270 - accuracy: 0.4439 - val_loss: 14.0768 - val_accura
cy: 0.4373
Epoch 114/200
391/391 - 6s - loss: 14.0058 - accuracy: 0.4443 - val_loss: 13.9581 - val_accura
cy: 0.4387
Epoch 115/200
391/391 - 6s - loss: 13.8857 - accuracy: 0.4450 - val_loss: 13.8379 - val_accura
cy: 0.4362
Epoch 116/200
391/391 - 6s - loss: 13.7665 - accuracy: 0.4449 - val_loss: 13.7192 - val_accura
cy: 0.4380
Epoch 117/200
391/391 - 6s - loss: 13.6492 - accuracy: 0.4445 - val_loss: 13.6026 - val_accura
cy: 0.4382
Epoch 118/200
391/391 - 6s - loss: 13.5330 - accuracy: 0.4460 - val_loss: 13.4871 - val_accura
cy: 0.4381
```

```
Epoch 119/200
391/391 - 6s - loss: 13.4180 - accuracy: 0.4460 - val_loss: 13.3719 - val_accura
cy: 0.4403
Epoch 120/200
391/391 - 6s - loss: 13.3040 - accuracy: 0.4467 - val_loss: 13.2589 - val_accura
cy: 0.4389
Epoch 121/200
391/391 - 6s - loss: 13.1913 - accuracy: 0.4465 - val_loss: 13.1469 - val_accura
cy: 0.4368
Epoch 122/200
391/391 - 6s - loss: 13.0801 - accuracy: 0.4465 - val_loss: 13.0376 - val_accura
cy: 0.4353
Epoch 123/200
391/391 - 5s - loss: 12.9704 - accuracy: 0.4466 - val_loss: 12.9271 - val_accura
cy: 0.4380
Epoch 124/200
391/391 - 6s - loss: 12.8611 - accuracy: 0.4473 - val_loss: 12.8182 - val_accura
cy: 0.4415
Epoch 125/200
391/391 - 6s - loss: 12.7532 - accuracy: 0.4476 - val_loss: 12.7120 - val_accura
cy: 0.4402
Epoch 126/200
391/391 - 6s - loss: 12.6461 - accuracy: 0.4481 - val_loss: 12.6046 - val_accura
cy: 0.4392
Epoch 127/200
391/391 - 6s - loss: 12.5397 - accuracy: 0.4472 - val_loss: 12.4989 - val_accura
cy: 0.4396
Epoch 128/200
391/391 - 6s - loss: 12.4350 - accuracy: 0.4484 - val_loss: 12.3950 - val_accura
cy: 0.4420
Epoch 129/200
391/391 - 6s - loss: 12.3316 - accuracy: 0.4476 - val_loss: 12.2923 - val_accura
cy: 0.4409
Epoch 130/200
391/391 - 5s - loss: 12.2292 - accuracy: 0.4480 - val_loss: 12.1902 - val_accura
cy: 0.4423
Epoch 131/200
391/391 - 6s - loss: 12.1278 - accuracy: 0.4489 - val_loss: 12.0897 - val_accura
cy: 0.4407
Epoch 132/200
391/391 - 6s - loss: 12.0275 - accuracy: 0.4484 - val_loss: 11.9899 - val_accura
cy: 0.4409
Epoch 133/200
391/391 - 6s - loss: 11.9283 - accuracy: 0.4490 - val_loss: 11.8909 - val_accura
cy: 0.4424
Epoch 134/200
391/391 - 6s - loss: 11.8295 - accuracy: 0.4496 - val_loss: 11.7927 - val_accura
cy: 0.4419
Epoch 135/200
391/391 - 6s - loss: 11.7318 - accuracy: 0.4502 - val_loss: 11.6963 - val_accura
cy: 0.4409
Epoch 136/200
391/391 - 6s - loss: 11.6354 - accuracy: 0.4504 - val_loss: 11.6003 - val_accura
cy: 0.4398
Epoch 137/200
391/391 - 6s - loss: 11.5399 - accuracy: 0.4501 - val_loss: 11.5055 - val_accura
cy: 0.4410
Epoch 138/200
391/391 - 6s - loss: 11.4458 - accuracy: 0.4501 - val_loss: 11.4116 - val_accura
```

```
cy: 0.4427
Epoch 139/200
391/391 - 6s - loss: 11.3526 - accuracy: 0.4498 - val_loss: 11.3186 - val_accura
cy: 0.4402
Epoch 140/200
391/391 - 5s - loss: 11.2602 - accuracy: 0.4502 - val_loss: 11.2272 - val_accura
cy: 0.4427
Epoch 141/200
391/391 - 6s - loss: 11.1690 - accuracy: 0.4508 - val_loss: 11.1358 - val_accura
cy: 0.4417
Epoch 142/200
391/391 - 6s - loss: 11.0786 - accuracy: 0.4507 - val_loss: 11.0460 - val_accura
cy: 0.4442
Epoch 143/200
391/391 - 6s - loss: 10.9889 - accuracy: 0.4507 - val_loss: 10.9568 - val_accura
cy: 0.4440
Epoch 144/200
391/391 - 6s - loss: 10.9004 - accuracy: 0.4516 - val_loss: 10.8693 - val_accura
cy: 0.4419
Epoch 145/200
391/391 - 6s - loss: 10.8129 - accuracy: 0.4512 - val_loss: 10.7820 - val_accura
cy: 0.4425
Epoch 146/200
391/391 - 6s - loss: 10.7265 - accuracy: 0.4524 - val_loss: 10.6961 - val_accura
cy: 0.4450
Epoch 147/200
391/391 - 6s - loss: 10.6406 - accuracy: 0.4523 - val_loss: 10.6109 - val_accura
cy: 0.4431
Epoch 148/200
391/391 - 6s - loss: 10.5556 - accuracy: 0.4531 - val_loss: 10.5265 - val_accura
cy: 0.4425
Epoch 149/200
391/391 - 6s - loss: 10.4714 - accuracy: 0.4522 - val_loss: 10.4427 - val_accura
cy: 0.4435
Epoch 150/200
391/391 - 6s - loss: 10.3881 - accuracy: 0.4525 - val_loss: 10.3597 - val_accura
cy: 0.4464
Epoch 151/200
391/391 - 6s - loss: 10.3058 - accuracy: 0.4537 - val_loss: 10.2788 - val_accura
cy: 0.4461
Epoch 152/200
391/391 - 6s - loss: 10.2244 - accuracy: 0.4531 - val_loss: 10.1965 - val_accura
cy: 0.4423
Epoch 153/200
391/391 - 6s - loss: 10.1437 - accuracy: 0.4530 - val_loss: 10.1169 - val_accura
cy: 0.4429
Epoch 154/200
391/391 - 6s - loss: 10.0640 - accuracy: 0.4536 - val_loss: 10.0372 - val_accura
cy: 0.4449
Epoch 155/200
391/391 - 6s - loss: 9.9850 - accuracy: 0.4547 - val_loss: 9.9591 - val_accurac
y: 0.4442
Epoch 156/200
391/391 - 6s - loss: 9.9068 - accuracy: 0.4543 - val_loss: 9.8820 - val_accurac
y: 0.4439
Epoch 157/200
391/391 - 6s - loss: 9.8302 - accuracy: 0.4525 - val_loss: 9.8052 - val_accurac
y: 0.4466
Epoch 158/200
```

```
391/391 - 6s - loss: 9.7540 - accuracy: 0.4543 - val_loss: 9.7299 - val_accurac
y: 0.4466
Epoch 159/200
391/391 - 6s - loss: 9.6782 - accuracy: 0.4541 - val_loss: 9.6540 - val_accurac
y: 0.4461
Epoch 160/200
391/391 - 6s - loss: 9.6031 - accuracy: 0.4549 - val_loss: 9.5798 - val_accurac
y: 0.4433
Epoch 161/200
391/391 - 6s - loss: 9.5289 - accuracy: 0.4539 - val_loss: 9.5054 - val_accurac
y: 0.4470
Epoch 162/200
391/391 - 6s - loss: 9.4556 - accuracy: 0.4553 - val_loss: 9.4326 - val_accurac
y: 0.4446
Epoch 163/200
391/391 - 6s - loss: 9.3828 - accuracy: 0.4552 - val_loss: 9.3604 - val_accurac
y: 0.4476
Epoch 164/200
391/391 - 5s - loss: 9.3111 - accuracy: 0.4558 - val_loss: 9.2897 - val_accurac
y: 0.4450
Epoch 165/200
391/391 - 5s - loss: 9.2398 - accuracy: 0.4549 - val_loss: 9.2178 - val_accurac
y: 0.4463
Epoch 166/200
391/391 - 3s - loss: 9.1689 - accuracy: 0.4558 - val_loss: 9.1476 - val_accurac
y: 0.4464
Epoch 167/200
391/391 - 3s - loss: 9.0991 - accuracy: 0.4564 - val_loss: 9.0781 - val_accurac
y: 0.4480
Epoch 168/200
391/391 - 3s - loss: 9.0300 - accuracy: 0.4558 - val_loss: 9.0090 - val_accurac
y: 0.4466
Epoch 169/200
391/391 - 3s - loss: 8.9611 - accuracy: 0.4550 - val_loss: 8.9407 - val_accurac
y: 0.4476
Epoch 170/200
391/391 - 3s - loss: 8.8933 - accuracy: 0.4563 - val_loss: 8.8733 - val_accurac
y: 0.4479
Epoch 171/200
391/391 - 4s - loss: 8.8259 - accuracy: 0.4573 - val_loss: 8.8069 - val_accurac
y: 0.4464
Epoch 172/200
391/391 - 3s - loss: 8.7594 - accuracy: 0.4561 - val_loss: 8.7401 - val_accurac
y: 0.4476
Epoch 173/200
391/391 - 3s - loss: 8.6935 - accuracy: 0.4566 - val_loss: 8.6744 - val_accurac
y: 0.4487
Epoch 174/200
391/391 - 3s - loss: 8.6286 - accuracy: 0.4574 - val_loss: 8.6104 - val_accurac
y: 0.4497
Epoch 175/200
391/391 - 4s - loss: 8.5640 - accuracy: 0.4575 - val_loss: 8.5461 - val_accurac
y: 0.4486
Epoch 176/200
391/391 - 4s - loss: 8.5002 - accuracy: 0.4581 - val_loss: 8.4822 - val_accurac
y: 0.4491
Epoch 177/200
391/391 - 3s - loss: 8.4370 - accuracy: 0.4579 - val_loss: 8.4199 - val_accurac
y: 0.4503
```

```
Epoch 178/200
391/391 - 4s - loss: 8.3745 - accuracy: 0.4573 - val_loss: 8.3583 - val_accurac
y: 0.4467
Epoch 179/200
391/391 - 4s - loss: 8.3130 - accuracy: 0.4573 - val_loss: 8.2959 - val_accurac
y: 0.4508
Epoch 180/200
391/391 - 4s - loss: 8.2515 - accuracy: 0.4574 - val_loss: 8.2358 - val_accurac
y: 0.4500
Epoch 181/200
391/391 - 3s - loss: 8.1910 - accuracy: 0.4575 - val_loss: 8.1751 - val_accurac
y: 0.4494
Epoch 182/200
391/391 - 4s - loss: 8.1313 - accuracy: 0.4585 - val_loss: 8.1168 - val_accurac
y: 0.4493
Epoch 183/200
391/391 - 3s - loss: 8.0719 - accuracy: 0.4575 - val_loss: 8.0571 - val_accurac
y: 0.4492
Epoch 184/200
391/391 - 3s - loss: 8.0128 - accuracy: 0.4584 - val_loss: 7.9977 - val_accurac
y: 0.4494
Epoch 185/200
391/391 - 3s - loss: 7.9544 - accuracy: 0.4583 - val_loss: 7.9399 - val_accurac
y: 0.4500
Epoch 186/200
391/391 - 3s - loss: 7.8969 - accuracy: 0.4592 - val_loss: 7.8834 - val_accurac
y: 0.4479
Epoch 187/200
391/391 - 4s - loss: 7.8402 - accuracy: 0.4590 - val_loss: 7.8261 - val_accurac
y: 0.4519
Epoch 188/200
391/391 - 4s - loss: 7.7839 - accuracy: 0.4586 - val_loss: 7.7706 - val_accurac
y: 0.4519
Epoch 189/200
391/391 - 3s - loss: 7.7280 - accuracy: 0.4601 - val_loss: 7.7147 - val_accurac
y: 0.4501
Epoch 190/200
391/391 - 4s - loss: 7.6726 - accuracy: 0.4591 - val_loss: 7.6594 - val_accurac
y: 0.4503
Epoch 191/200
391/391 - 4s - loss: 7.6177 - accuracy: 0.4594 - val_loss: 7.6045 - val_accurac
y: 0.4516
Epoch 192/200
391/391 - 4s - loss: 7.5630 - accuracy: 0.4591 - val_loss: 7.5512 - val_accurac
y: 0.4503
Epoch 193/200
391/391 - 3s - loss: 7.5095 - accuracy: 0.4594 - val_loss: 7.4973 - val_accurac
y: 0.4516
Epoch 194/200
391/391 - 4s - loss: 7.4565 - accuracy: 0.4607 - val_loss: 7.4447 - val_accurac
y: 0.4517
Epoch 195/200
391/391 - 3s - loss: 7.4040 - accuracy: 0.4595 - val_loss: 7.3925 - val_accurac
y: 0.4511
Epoch 196/200
391/391 - 4s - loss: 7.3521 - accuracy: 0.4604 - val_loss: 7.3416 - val_accurac
y: 0.4521
Epoch 197/200
391/391 - 3s - loss: 7.3006 - accuracy: 0.4599 - val_loss: 7.2895 - val_accurac
```

```
y: 0.4534
Epoch 198/200
391/391 - 3s - loss: 7.2495 - accuracy: 0.4615 - val_loss: 7.2396 - val_accurac
y: 0.4507
Epoch 199/200
391/391 - 3s - loss: 7.1992 - accuracy: 0.4605 - val_loss: 7.1893 - val_accurac
y: 0.4511
Epoch 200/200
391/391 - 3s - loss: 7.1491 - accuracy: 0.4602 - val_loss: 7.1394 - val_accurac
y: 0.4537
```

Adam

In [21]:
```
model_adam=create_model()
model_adam.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
history_adam = model_adam.fit(X_train, y_train, batch_size=128, epochs=200, validation_data=(X_test,y_test),shuffle=True,verbose=2)
```

```
Epoch 1/200
391/391 - 4s - loss: 7.2912 - accuracy: 0.3047 - val_loss: 2.4559 - val_accurac
y: 0.2939
Epoch 2/200
391/391 - 3s - loss: 2.1188 - accuracy: 0.3571 - val_loss: 2.0510 - val_accurac
y: 0.3199
Epoch 3/200
391/391 - 3s - loss: 1.9158 - accuracy: 0.3667 - val_loss: 1.8877 - val_accurac
y: 0.3676
Epoch 4/200
391/391 - 3s - loss: 1.8566 - accuracy: 0.3760 - val_loss: 1.8920 - val_accurac
y: 0.3592
Epoch 5/200
391/391 - 3s - loss: 1.8315 - accuracy: 0.3822 - val_loss: 1.7849 - val_accurac
y: 0.4011
Epoch 6/200
391/391 - 3s - loss: 1.7969 - accuracy: 0.3961 - val_loss: 1.7698 - val_accurac
y: 0.3941
Epoch 7/200
391/391 - 3s - loss: 1.7801 - accuracy: 0.4031 - val_loss: 1.7355 - val_accurac
y: 0.4220
Epoch 8/200
391/391 - 3s - loss: 1.7699 - accuracy: 0.4073 - val_loss: 1.8007 - val_accurac
y: 0.3931
Epoch 9/200
391/391 - 3s - loss: 1.7596 - accuracy: 0.4091 - val_loss: 1.7220 - val_accurac
y: 0.4227
Epoch 10/200
391/391 - 3s - loss: 1.7705 - accuracy: 0.4066 - val_loss: 1.7981 - val_accurac
y: 0.3946
Epoch 11/200
391/391 - 3s - loss: 1.7529 - accuracy: 0.4132 - val_loss: 1.7720 - val_accurac
y: 0.4006
Epoch 12/200
391/391 - 3s - loss: 1.7432 - accuracy: 0.4158 - val_loss: 1.7668 - val_accurac
y: 0.4056
Epoch 13/200
391/391 - 3s - loss: 1.7314 - accuracy: 0.4216 - val_loss: 1.7031 - val_accurac
y: 0.4367
Epoch 14/200
391/391 - 3s - loss: 1.7329 - accuracy: 0.4197 - val_loss: 1.7024 - val_accurac
y: 0.4387
Epoch 15/200
391/391 - 3s - loss: 1.7266 - accuracy: 0.4184 - val_loss: 1.6956 - val_accurac
y: 0.4345
Epoch 16/200
391/391 - 3s - loss: 1.7219 - accuracy: 0.4220 - val_loss: 1.7148 - val_accurac
y: 0.4308
Epoch 17/200
391/391 - 3s - loss: 1.7211 - accuracy: 0.4224 - val_loss: 1.7024 - val_accurac
y: 0.4317
Epoch 18/200
391/391 - 3s - loss: 1.7159 - accuracy: 0.4268 - val_loss: 1.6871 - val_accurac
y: 0.4379
Epoch 19/200
391/391 - 3s - loss: 1.7139 - accuracy: 0.4270 - val_loss: 1.7124 - val_accurac
y: 0.4262
Epoch 20/200
391/391 - 3s - loss: 1.7027 - accuracy: 0.4300 - val_loss: 1.7077 - val_accurac
```

```
y: 0.4269
Epoch 21/200
391/391 - 3s - loss: 1.7120 - accuracy: 0.4256 - val_loss: 1.6958 - val_accurac
y: 0.4324
Epoch 22/200
391/391 - 3s - loss: 1.6926 - accuracy: 0.4336 - val_loss: 1.6782 - val_accurac
y: 0.4353
Epoch 23/200
391/391 - 3s - loss: 1.6921 - accuracy: 0.4328 - val_loss: 1.7320 - val_accurac
y: 0.4147
Epoch 24/200
391/391 - 3s - loss: 1.6917 - accuracy: 0.4343 - val_loss: 1.6668 - val_accurac
y: 0.4496
Epoch 25/200
391/391 - 3s - loss: 1.6885 - accuracy: 0.4337 - val_loss: 1.7185 - val_accurac
y: 0.4230
Epoch 26/200
391/391 - 3s - loss: 1.6921 - accuracy: 0.4306 - val_loss: 1.6820 - val_accurac
y: 0.4432
Epoch 27/200
391/391 - 3s - loss: 1.6822 - accuracy: 0.4359 - val_loss: 1.6923 - val_accurac
y: 0.4289
Epoch 28/200
391/391 - 3s - loss: 1.6842 - accuracy: 0.4353 - val_loss: 1.6695 - val_accurac
y: 0.4374
Epoch 29/200
391/391 - 3s - loss: 1.6805 - accuracy: 0.4379 - val_loss: 1.6452 - val_accurac
y: 0.4530
Epoch 30/200
391/391 - 3s - loss: 1.6799 - accuracy: 0.4373 - val_loss: 1.8377 - val_accurac
y: 0.3764
Epoch 31/200
391/391 - 3s - loss: 1.6741 - accuracy: 0.4371 - val_loss: 1.7015 - val_accurac
y: 0.4270
Epoch 32/200
391/391 - 3s - loss: 1.6729 - accuracy: 0.4408 - val_loss: 1.6519 - val_accurac
y: 0.4469
Epoch 33/200
391/391 - 3s - loss: 1.6777 - accuracy: 0.4341 - val_loss: 1.7349 - val_accurac
y: 0.4150
Epoch 34/200
391/391 - 3s - loss: 1.6855 - accuracy: 0.4335 - val_loss: 1.7552 - val_accurac
y: 0.4107
Epoch 35/200
391/391 - 3s - loss: 1.6850 - accuracy: 0.4331 - val_loss: 1.6727 - val_accurac
y: 0.4398
Epoch 36/200
391/391 - 3s - loss: 1.6764 - accuracy: 0.4373 - val_loss: 1.6863 - val_accurac
y: 0.4379
Epoch 37/200
391/391 - 3s - loss: 1.6818 - accuracy: 0.4333 - val_loss: 1.7067 - val_accurac
y: 0.4272
Epoch 38/200
391/391 - 3s - loss: 1.6663 - accuracy: 0.4406 - val_loss: 1.6730 - val_accurac
y: 0.4502
Epoch 39/200
391/391 - 3s - loss: 1.6717 - accuracy: 0.4389 - val_loss: 1.7356 - val_accurac
y: 0.4260
Epoch 40/200
```

```
391/391 - 3s - loss: 1.6691 - accuracy: 0.4411 - val_loss: 1.7124 - val_accurac
y: 0.4248
Epoch 41/200
391/391 - 3s - loss: 1.6683 - accuracy: 0.4385 - val_loss: 1.6519 - val_accurac
y: 0.4451
Epoch 42/200
391/391 - 3s - loss: 1.6721 - accuracy: 0.4386 - val_loss: 1.6788 - val_accurac
y: 0.4355
Epoch 43/200
391/391 - 3s - loss: 1.6759 - accuracy: 0.4362 - val_loss: 1.7011 - val_accurac
y: 0.4299
Epoch 44/200
391/391 - 3s - loss: 1.6704 - accuracy: 0.4392 - val_loss: 1.6969 - val_accurac
y: 0.4348
Epoch 45/200
391/391 - 3s - loss: 1.6720 - accuracy: 0.4403 - val_loss: 1.7276 - val_accurac
y: 0.4170
Epoch 46/200
391/391 - 3s - loss: 1.6644 - accuracy: 0.4409 - val_loss: 1.6611 - val_accurac
y: 0.4460
Epoch 47/200
391/391 - 3s - loss: 1.6598 - accuracy: 0.4417 - val_loss: 1.7103 - val_accurac
y: 0.4270
Epoch 48/200
391/391 - 3s - loss: 1.6647 - accuracy: 0.4398 - val_loss: 1.6644 - val_accurac
y: 0.4372
Epoch 49/200
391/391 - 3s - loss: 1.6644 - accuracy: 0.4408 - val_loss: 1.7431 - val_accurac
y: 0.4097
Epoch 50/200
391/391 - 3s - loss: 1.6598 - accuracy: 0.4438 - val_loss: 1.6775 - val_accurac
y: 0.4254
Epoch 51/200
391/391 - 3s - loss: 1.6642 - accuracy: 0.4398 - val_loss: 1.6581 - val_accurac
y: 0.4432
Epoch 52/200
391/391 - 3s - loss: 1.6529 - accuracy: 0.4467 - val_loss: 1.6563 - val_accurac
y: 0.4474
Epoch 53/200
391/391 - 3s - loss: 1.6620 - accuracy: 0.4410 - val_loss: 1.6553 - val_accurac
y: 0.4483
Epoch 54/200
391/391 - 3s - loss: 1.6602 - accuracy: 0.4434 - val_loss: 1.6616 - val_accurac
y: 0.4441
Epoch 55/200
391/391 - 3s - loss: 1.6566 - accuracy: 0.4441 - val_loss: 1.6827 - val_accurac
y: 0.4445
Epoch 56/200
391/391 - 3s - loss: 1.6692 - accuracy: 0.4392 - val_loss: 1.6808 - val_accurac
y: 0.4391
Epoch 57/200
391/391 - 3s - loss: 1.6617 - accuracy: 0.4430 - val_loss: 1.6515 - val_accurac
y: 0.4502
Epoch 58/200
391/391 - 3s - loss: 1.6640 - accuracy: 0.4419 - val_loss: 1.6425 - val_accurac
y: 0.4556
Epoch 59/200
391/391 - 3s - loss: 1.6542 - accuracy: 0.4454 - val_loss: 1.6596 - val_accurac
y: 0.4449
```

```
Epoch 60/200
391/391 - 3s - loss: 1.6595 - accuracy: 0.4407 - val_loss: 1.6454 - val_accurac
y: 0.4540
Epoch 61/200
391/391 - 3s - loss: 1.6568 - accuracy: 0.4435 - val_loss: 1.6506 - val_accurac
y: 0.4513
Epoch 62/200
391/391 - 3s - loss: 1.6586 - accuracy: 0.4419 - val_loss: 1.6407 - val_accurac
y: 0.4502
Epoch 63/200
391/391 - 3s - loss: 1.6619 - accuracy: 0.4445 - val_loss: 1.6970 - val_accurac
y: 0.4375
Epoch 64/200
391/391 - 3s - loss: 1.6617 - accuracy: 0.4421 - val_loss: 1.6474 - val_accurac
y: 0.4559
Epoch 65/200
391/391 - 3s - loss: 1.6548 - accuracy: 0.4433 - val_loss: 1.6943 - val_accurac
y: 0.4282
Epoch 66/200
391/391 - 3s - loss: 1.6643 - accuracy: 0.4419 - val_loss: 1.6387 - val_accurac
y: 0.4522
Epoch 67/200
391/391 - 3s - loss: 1.6495 - accuracy: 0.4460 - val_loss: 1.6454 - val_accurac
y: 0.4478
Epoch 68/200
391/391 - 3s - loss: 1.6623 - accuracy: 0.4436 - val_loss: 1.7547 - val_accurac
y: 0.4092
Epoch 69/200
391/391 - 3s - loss: 1.6527 - accuracy: 0.4448 - val_loss: 1.6522 - val_accurac
y: 0.4484
Epoch 70/200
391/391 - 3s - loss: 1.6592 - accuracy: 0.4425 - val_loss: 1.6631 - val_accurac
y: 0.4454
Epoch 71/200
391/391 - 3s - loss: 1.6533 - accuracy: 0.4451 - val_loss: 1.6380 - val_accurac
y: 0.4496
Epoch 72/200
391/391 - 3s - loss: 1.6568 - accuracy: 0.4429 - val_loss: 1.6292 - val_accurac
y: 0.4594
Epoch 73/200
391/391 - 3s - loss: 1.6519 - accuracy: 0.4444 - val_loss: 1.6784 - val_accurac
y: 0.4441
Epoch 74/200
391/391 - 3s - loss: 1.6511 - accuracy: 0.4429 - val_loss: 1.6696 - val_accurac
y: 0.4388
Epoch 75/200
391/391 - 3s - loss: 1.6499 - accuracy: 0.4475 - val_loss: 1.6375 - val_accurac
y: 0.4550
Epoch 76/200
391/391 - 3s - loss: 1.6434 - accuracy: 0.4489 - val_loss: 1.6301 - val_accurac
y: 0.4631
Epoch 77/200
391/391 - 3s - loss: 1.6451 - accuracy: 0.4478 - val_loss: 1.7003 - val_accurac
y: 0.4301
Epoch 78/200
391/391 - 3s - loss: 1.6497 - accuracy: 0.4478 - val_loss: 1.6494 - val_accurac
y: 0.4487
Epoch 79/200
391/391 - 3s - loss: 1.6434 - accuracy: 0.4489 - val_loss: 1.6446 - val_accurac
```

```
y: 0.4527
Epoch 80/200
391/391 - 3s - loss: 1.6434 - accuracy: 0.4487 - val_loss: 1.6683 - val_accurac
y: 0.4414
Epoch 81/200
391/391 - 3s - loss: 1.6549 - accuracy: 0.4441 - val_loss: 1.6683 - val_accurac
y: 0.4415
Epoch 82/200
391/391 - 3s - loss: 1.6523 - accuracy: 0.4437 - val_loss: 1.6718 - val_accurac
y: 0.4467
Epoch 83/200
391/391 - 3s - loss: 1.6501 - accuracy: 0.4464 - val_loss: 1.6613 - val_accurac
y: 0.4463
Epoch 84/200
391/391 - 3s - loss: 1.6482 - accuracy: 0.4488 - val_loss: 1.6826 - val_accurac
y: 0.4395
Epoch 85/200
391/391 - 3s - loss: 1.6482 - accuracy: 0.4466 - val_loss: 1.6715 - val_accurac
y: 0.4438
Epoch 86/200
391/391 - 3s - loss: 1.6519 - accuracy: 0.4478 - val_loss: 1.6470 - val_accurac
y: 0.4537
Epoch 87/200
391/391 - 3s - loss: 1.6462 - accuracy: 0.4473 - val_loss: 1.6399 - val_accurac
y: 0.4482
Epoch 88/200
391/391 - 3s - loss: 1.6477 - accuracy: 0.4464 - val_loss: 1.6410 - val_accurac
y: 0.4551
Epoch 89/200
391/391 - 3s - loss: 1.6454 - accuracy: 0.4497 - val_loss: 1.6481 - val_accurac
y: 0.4429
Epoch 90/200
391/391 - 3s - loss: 1.6507 - accuracy: 0.4470 - val_loss: 1.6062 - val_accurac
y: 0.4720
Epoch 91/200
391/391 - 3s - loss: 1.6408 - accuracy: 0.4499 - val_loss: 1.7444 - val_accurac
y: 0.4147
Epoch 92/200
391/391 - 3s - loss: 1.6714 - accuracy: 0.4373 - val_loss: 1.6982 - val_accurac
y: 0.4281
Epoch 93/200
391/391 - 3s - loss: 1.6515 - accuracy: 0.4460 - val_loss: 1.6715 - val_accurac
y: 0.4449
Epoch 94/200
391/391 - 3s - loss: 1.6432 - accuracy: 0.4453 - val_loss: 1.6504 - val_accurac
y: 0.4488
Epoch 95/200
391/391 - 3s - loss: 1.6427 - accuracy: 0.4486 - val_loss: 1.6500 - val_accurac
y: 0.4464
Epoch 96/200
391/391 - 3s - loss: 1.6396 - accuracy: 0.4493 - val_loss: 1.6988 - val_accurac
y: 0.4274
Epoch 97/200
391/391 - 3s - loss: 1.6474 - accuracy: 0.4464 - val_loss: 1.7100 - val_accurac
y: 0.4311
Epoch 98/200
391/391 - 3s - loss: 1.6543 - accuracy: 0.4423 - val_loss: 1.6464 - val_accurac
y: 0.4448
Epoch 99/200
```

```
391/391 - 3s - loss: 1.6514 - accuracy: 0.4475 - val_loss: 1.6110 - val_accurac
y: 0.4659
Epoch 100/200
391/391 - 3s - loss: 1.6434 - accuracy: 0.4482 - val_loss: 1.6494 - val_accurac
y: 0.4535
Epoch 101/200
391/391 - 3s - loss: 1.6481 - accuracy: 0.4459 - val_loss: 1.7366 - val_accurac
y: 0.4220
Epoch 102/200
391/391 - 3s - loss: 1.6499 - accuracy: 0.4489 - val_loss: 1.8381 - val_accurac
y: 0.3856
Epoch 103/200
391/391 - 3s - loss: 1.6444 - accuracy: 0.4472 - val_loss: 1.6526 - val_accurac
y: 0.4465
Epoch 104/200
391/391 - 3s - loss: 1.6521 - accuracy: 0.4460 - val_loss: 1.6582 - val_accurac
y: 0.4406
Epoch 105/200
391/391 - 3s - loss: 1.6437 - accuracy: 0.4496 - val_loss: 1.6888 - val_accurac
y: 0.4381
Epoch 106/200
391/391 - 3s - loss: 1.6424 - accuracy: 0.4478 - val_loss: 1.7045 - val_accurac
y: 0.4311
Epoch 107/200
391/391 - 3s - loss: 1.6502 - accuracy: 0.4446 - val_loss: 1.6146 - val_accurac
y: 0.4632
Epoch 108/200
391/391 - 3s - loss: 1.6392 - accuracy: 0.4500 - val_loss: 1.6517 - val_accurac
y: 0.4481
Epoch 109/200
391/391 - 3s - loss: 1.6525 - accuracy: 0.4455 - val_loss: 1.6424 - val_accurac
y: 0.4531
Epoch 110/200
391/391 - 3s - loss: 1.6402 - accuracy: 0.4491 - val_loss: 1.6015 - val_accurac
y: 0.4644
Epoch 111/200
391/391 - 3s - loss: 1.6427 - accuracy: 0.4505 - val_loss: 1.6738 - val_accurac
y: 0.4321
Epoch 112/200
391/391 - 3s - loss: 1.6443 - accuracy: 0.4466 - val_loss: 1.6487 - val_accurac
y: 0.4540
Epoch 113/200
391/391 - 3s - loss: 1.6387 - accuracy: 0.4490 - val_loss: 1.5991 - val_accurac
y: 0.4639
Epoch 114/200
391/391 - 3s - loss: 1.6448 - accuracy: 0.4445 - val_loss: 1.6669 - val_accurac
y: 0.4463
Epoch 115/200
391/391 - 3s - loss: 1.6469 - accuracy: 0.4473 - val_loss: 1.6282 - val_accurac
y: 0.4543
Epoch 116/200
391/391 - 3s - loss: 1.6376 - accuracy: 0.4496 - val_loss: 1.6346 - val_accurac
y: 0.4598
Epoch 117/200
391/391 - 3s - loss: 1.6480 - accuracy: 0.4458 - val_loss: 1.6517 - val_accurac
y: 0.4523
Epoch 118/200
391/391 - 3s - loss: 1.6483 - accuracy: 0.4467 - val_loss: 1.7274 - val_accurac
y: 0.4230
```

```
Epoch 119/200
391/391 - 3s - loss: 1.6392 - accuracy: 0.4509 - val_loss: 1.7058 - val_accurac
y: 0.4284
Epoch 120/200
391/391 - 3s - loss: 1.6412 - accuracy: 0.4474 - val_loss: 1.6350 - val_accurac
y: 0.4556
Epoch 121/200
391/391 - 3s - loss: 1.6417 - accuracy: 0.4485 - val_loss: 1.6737 - val_accurac
y: 0.4397
Epoch 122/200
391/391 - 3s - loss: 1.6313 - accuracy: 0.4521 - val_loss: 1.6511 - val_accurac
y: 0.4464
Epoch 123/200
391/391 - 3s - loss: 1.6404 - accuracy: 0.4503 - val_loss: 1.6821 - val_accurac
y: 0.4335
Epoch 124/200
391/391 - 3s - loss: 1.6318 - accuracy: 0.4520 - val_loss: 1.6216 - val_accurac
y: 0.4636
Epoch 125/200
391/391 - 3s - loss: 1.6441 - accuracy: 0.4485 - val_loss: 1.6564 - val_accurac
y: 0.4466
Epoch 126/200
391/391 - 3s - loss: 1.6472 - accuracy: 0.4479 - val_loss: 1.6894 - val_accurac
y: 0.4360
Epoch 127/200
391/391 - 3s - loss: 1.6390 - accuracy: 0.4509 - val_loss: 1.5968 - val_accurac
y: 0.4690
Epoch 128/200
391/391 - 3s - loss: 1.6389 - accuracy: 0.4472 - val_loss: 1.6040 - val_accurac
y: 0.4700
Epoch 129/200
391/391 - 3s - loss: 1.6422 - accuracy: 0.4471 - val_loss: 1.6214 - val_accurac
y: 0.4631
Epoch 130/200
391/391 - 3s - loss: 1.6337 - accuracy: 0.4518 - val_loss: 1.6913 - val_accurac
y: 0.4362
Epoch 131/200
391/391 - 3s - loss: 1.6355 - accuracy: 0.4505 - val_loss: 1.6410 - val_accurac
y: 0.4537
Epoch 132/200
391/391 - 3s - loss: 1.6424 - accuracy: 0.4469 - val_loss: 1.6728 - val_accurac
y: 0.4329
Epoch 133/200
391/391 - 3s - loss: 1.6390 - accuracy: 0.4497 - val_loss: 1.6375 - val_accurac
y: 0.4540
Epoch 134/200
391/391 - 3s - loss: 1.6346 - accuracy: 0.4488 - val_loss: 1.6883 - val_accurac
y: 0.4346
Epoch 135/200
391/391 - 3s - loss: 1.6373 - accuracy: 0.4527 - val_loss: 1.6253 - val_accurac
y: 0.4582
Epoch 136/200
391/391 - 3s - loss: 1.6411 - accuracy: 0.4514 - val_loss: 1.6284 - val_accurac
y: 0.4526
Epoch 137/200
391/391 - 3s - loss: 1.6630 - accuracy: 0.4397 - val_loss: 1.6268 - val_accurac
y: 0.4575
Epoch 138/200
391/391 - 3s - loss: 1.6282 - accuracy: 0.4534 - val_loss: 1.6482 - val_accurac
```

```
y: 0.4472
Epoch 139/200
391/391 - 3s - loss: 1.6357 - accuracy: 0.4518 - val_loss: 1.6846 - val_accurac
y: 0.4372
Epoch 140/200
391/391 - 3s - loss: 1.6377 - accuracy: 0.4501 - val_loss: 1.6175 - val_accurac
y: 0.4618
Epoch 141/200
391/391 - 3s - loss: 1.6358 - accuracy: 0.4510 - val_loss: 1.6358 - val_accurac
y: 0.4608
Epoch 142/200
391/391 - 3s - loss: 1.6303 - accuracy: 0.4540 - val_loss: 1.6234 - val_accurac
y: 0.4600
Epoch 143/200
391/391 - 3s - loss: 1.6428 - accuracy: 0.4482 - val_loss: 1.6532 - val_accurac
y: 0.4469
Epoch 144/200
391/391 - 3s - loss: 1.6385 - accuracy: 0.4516 - val_loss: 1.6082 - val_accurac
y: 0.4713
Epoch 145/200
391/391 - 3s - loss: 1.6428 - accuracy: 0.4499 - val_loss: 1.6522 - val_accurac
y: 0.4470
Epoch 146/200
391/391 - 3s - loss: 1.6295 - accuracy: 0.4556 - val_loss: 1.6729 - val_accurac
y: 0.4418
Epoch 147/200
391/391 - 3s - loss: 1.6334 - accuracy: 0.4539 - val_loss: 1.6929 - val_accurac
y: 0.4364
Epoch 148/200
391/391 - 3s - loss: 1.6318 - accuracy: 0.4525 - val_loss: 1.6741 - val_accurac
y: 0.4313
Epoch 149/200
391/391 - 3s - loss: 1.6444 - accuracy: 0.4491 - val_loss: 1.6147 - val_accurac
y: 0.4619
Epoch 150/200
391/391 - 3s - loss: 1.6413 - accuracy: 0.4493 - val_loss: 1.7043 - val_accurac
y: 0.4203
Epoch 151/200
391/391 - 3s - loss: 1.6352 - accuracy: 0.4517 - val_loss: 1.6575 - val_accurac
y: 0.4487
Epoch 152/200
391/391 - 3s - loss: 1.6396 - accuracy: 0.4523 - val_loss: 1.6150 - val_accurac
y: 0.4706
Epoch 153/200
391/391 - 3s - loss: 1.6339 - accuracy: 0.4527 - val_loss: 1.6689 - val_accurac
y: 0.4446
Epoch 154/200
391/391 - 3s - loss: 1.6485 - accuracy: 0.4483 - val_loss: 1.6547 - val_accurac
y: 0.4510
Epoch 155/200
391/391 - 3s - loss: 1.6313 - accuracy: 0.4529 - val_loss: 1.6134 - val_accurac
y: 0.4682
Epoch 156/200
391/391 - 3s - loss: 1.6398 - accuracy: 0.4507 - val_loss: 1.6203 - val_accurac
y: 0.4651
Epoch 157/200
391/391 - 3s - loss: 1.6328 - accuracy: 0.4517 - val_loss: 1.5988 - val_accurac
y: 0.4682
Epoch 158/200
```

```
391/391 - 3s - loss: 1.6339 - accuracy: 0.4536 - val_loss: 1.6869 - val_accurac
y: 0.4334
Epoch 159/200
391/391 - 3s - loss: 1.6311 - accuracy: 0.4544 - val_loss: 1.6057 - val_accurac
y: 0.4697
Epoch 160/200
391/391 - 3s - loss: 1.6312 - accuracy: 0.4523 - val_loss: 1.6405 - val_accurac
y: 0.4540
Epoch 161/200
391/391 - 3s - loss: 1.6338 - accuracy: 0.4518 - val_loss: 1.6974 - val_accurac
y: 0.4279
Epoch 162/200
391/391 - 3s - loss: 1.6354 - accuracy: 0.4519 - val_loss: 1.6383 - val_accurac
y: 0.4484
Epoch 163/200
391/391 - 3s - loss: 1.6329 - accuracy: 0.4499 - val_loss: 1.6599 - val_accurac
y: 0.4420
Epoch 164/200
391/391 - 3s - loss: 1.6279 - accuracy: 0.4516 - val_loss: 1.6863 - val_accurac
y: 0.4341
Epoch 165/200
391/391 - 3s - loss: 1.6336 - accuracy: 0.4521 - val_loss: 1.6251 - val_accurac
y: 0.4652
Epoch 166/200
391/391 - 3s - loss: 1.6400 - accuracy: 0.4505 - val_loss: 1.6218 - val_accurac
y: 0.4654
Epoch 167/200
391/391 - 3s - loss: 1.6356 - accuracy: 0.4520 - val_loss: 1.6311 - val_accurac
y: 0.4545
Epoch 168/200
391/391 - 3s - loss: 1.6371 - accuracy: 0.4514 - val_loss: 1.7240 - val_accurac
y: 0.4224
Epoch 169/200
391/391 - 3s - loss: 1.6351 - accuracy: 0.4506 - val_loss: 1.6386 - val_accurac
y: 0.4578
Epoch 170/200
391/391 - 3s - loss: 1.6362 - accuracy: 0.4491 - val_loss: 1.6801 - val_accurac
y: 0.4328
Epoch 171/200
391/391 - 3s - loss: 1.6400 - accuracy: 0.4500 - val_loss: 1.7163 - val_accurac
y: 0.4298
Epoch 172/200
391/391 - 3s - loss: 1.6228 - accuracy: 0.4544 - val_loss: 1.6401 - val_accurac
y: 0.4514
Epoch 173/200
391/391 - 3s - loss: 1.6434 - accuracy: 0.4456 - val_loss: 1.6260 - val_accurac
y: 0.4612
Epoch 174/200
391/391 - 3s - loss: 1.6264 - accuracy: 0.4521 - val_loss: 1.6299 - val_accurac
y: 0.4583
Epoch 175/200
391/391 - 3s - loss: 1.6328 - accuracy: 0.4510 - val_loss: 1.7282 - val_accurac
y: 0.4154
Epoch 176/200
391/391 - 3s - loss: 1.6217 - accuracy: 0.4549 - val_loss: 1.6464 - val_accurac
y: 0.4478
Epoch 177/200
391/391 - 3s - loss: 1.6351 - accuracy: 0.4510 - val_loss: 1.6847 - val_accurac
y: 0.4310
```

```
Epoch 178/200
391/391 - 3s - loss: 1.6388 - accuracy: 0.4501 - val_loss: 1.7154 - val_accurac
y: 0.4222
Epoch 179/200
391/391 - 3s - loss: 1.6359 - accuracy: 0.4522 - val_loss: 1.7282 - val_accurac
y: 0.4134
Epoch 180/200
391/391 - 3s - loss: 1.6294 - accuracy: 0.4528 - val_loss: 1.7774 - val_accurac
y: 0.4167
Epoch 181/200
391/391 - 3s - loss: 1.6242 - accuracy: 0.4550 - val_loss: 1.6867 - val_accurac
y: 0.4315
Epoch 182/200
391/391 - 3s - loss: 1.6362 - accuracy: 0.4517 - val_loss: 1.6431 - val_accurac
y: 0.4528
Epoch 183/200
391/391 - 3s - loss: 1.6402 - accuracy: 0.4505 - val_loss: 1.6904 - val_accurac
y: 0.4378
Epoch 184/200
391/391 - 3s - loss: 1.6364 - accuracy: 0.4521 - val_loss: 1.7376 - val_accurac
y: 0.4143
Epoch 185/200
391/391 - 3s - loss: 1.6300 - accuracy: 0.4521 - val_loss: 1.7385 - val_accurac
y: 0.4123
Epoch 186/200
391/391 - 3s - loss: 1.6393 - accuracy: 0.4502 - val_loss: 1.6186 - val_accurac
y: 0.4651
Epoch 187/200
391/391 - 3s - loss: 1.6287 - accuracy: 0.4548 - val_loss: 1.6790 - val_accurac
y: 0.4361
Epoch 188/200
391/391 - 3s - loss: 1.6296 - accuracy: 0.4549 - val_loss: 1.5895 - val_accurac
y: 0.4695
Epoch 189/200
391/391 - 3s - loss: 1.6299 - accuracy: 0.4510 - val_loss: 1.6442 - val_accurac
y: 0.4526
Epoch 190/200
391/391 - 3s - loss: 1.6318 - accuracy: 0.4532 - val_loss: 1.6512 - val_accurac
y: 0.4422
Epoch 191/200
391/391 - 3s - loss: 1.6302 - accuracy: 0.4536 - val_loss: 1.6002 - val_accurac
y: 0.4763
Epoch 192/200
391/391 - 3s - loss: 1.6365 - accuracy: 0.4511 - val_loss: 1.5999 - val_accurac
y: 0.4697
Epoch 193/200
391/391 - 3s - loss: 1.6293 - accuracy: 0.4538 - val_loss: 1.6001 - val_accurac
y: 0.4663
Epoch 194/200
391/391 - 3s - loss: 1.6329 - accuracy: 0.4534 - val_loss: 1.6668 - val_accurac
y: 0.4385
Epoch 195/200
391/391 - 3s - loss: 1.6310 - accuracy: 0.4536 - val_loss: 1.6672 - val_accurac
y: 0.4445
Epoch 196/200
391/391 - 3s - loss: 1.6215 - accuracy: 0.4549 - val_loss: 1.6945 - val_accurac
y: 0.4368
Epoch 197/200
391/391 - 3s - loss: 1.6357 - accuracy: 0.4553 - val_loss: 1.6936 - val_accurac
```

```
y: 0.4272
Epoch 198/200
391/391 - 3s - loss: 1.6323 - accuracy: 0.4524 - val_loss: 1.6829 - val_accurac
y: 0.4353
Epoch 199/200
391/391 - 3s - loss: 1.6356 - accuracy: 0.4507 - val_loss: 1.6542 - val_accurac
y: 0.4469
Epoch 200/200
391/391 - 3s - loss: 1.6286 - accuracy: 0.4525 - val_loss: 1.7330 - val_accurac
y: 0.4200
```

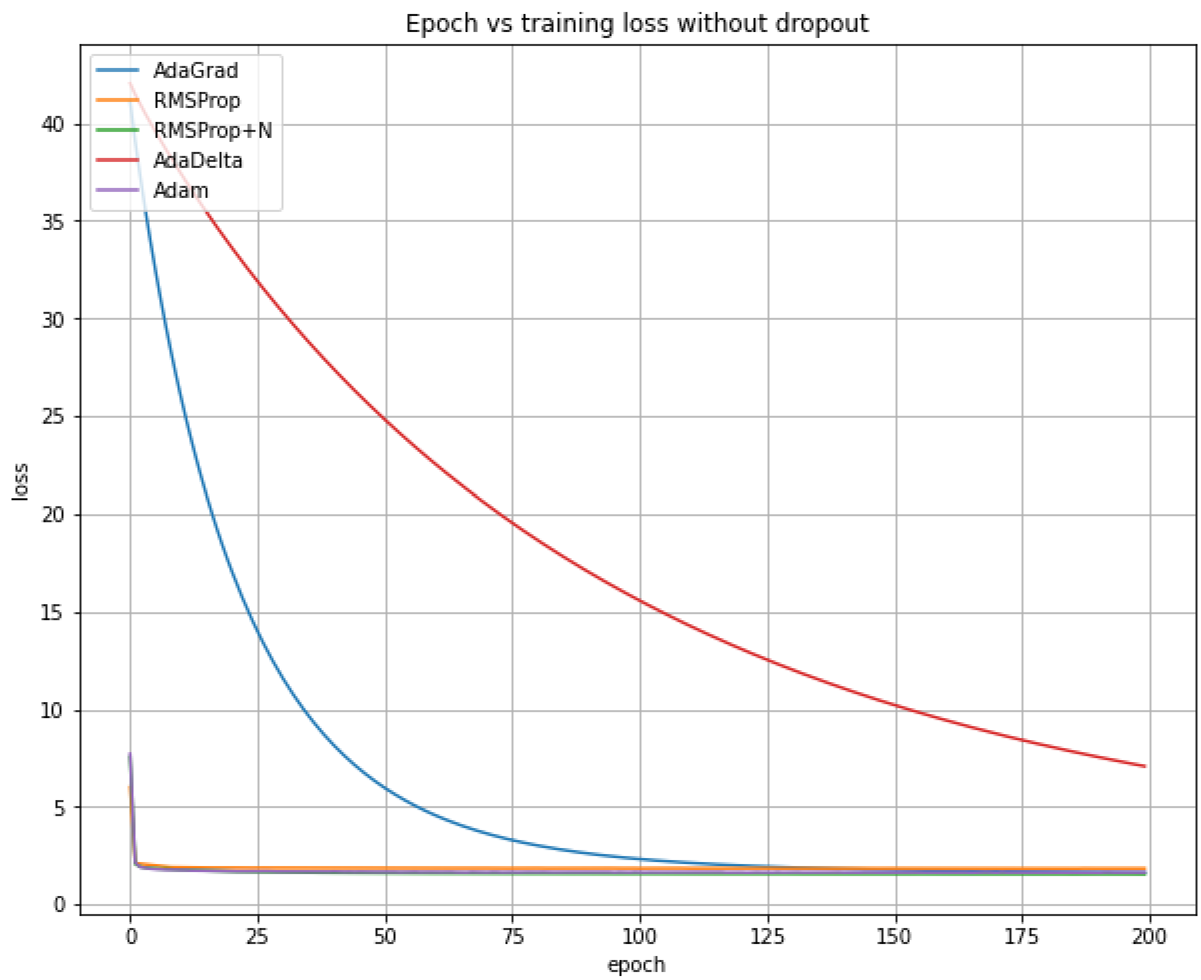
Lowest training loss

In [8]:

```
print("lowest training loss")
print("AdaGrad: {}".format(min(history_adagrad.history['loss'])))
print("RMSProp: {}".format(min(history_rmsprop.history['loss'])))
print("RMSProp+N: {}".format(min(history_nesterov.history['loss'])))
print("AdaDelta: {}".format(min(history_adadelta.history['loss'])))
print("Adam: {}".format(min(history_adam.history['loss'])))
```

```
lowest training loss
AdaGrad: 1.6244221925735474
RMSProp: 1.8377635478973389
RMSProp+N: 1.5503463745117188
AdaDelta: 7.079044342041016
Adam: 1.6213159561157227
```

In [9]:
```
from matplotlib import pyplot as plt
#Plotting
plt_loc='upper left'
fig = plt.figure(figsize=(10,8))
plt.plot(history_adagrad.history['loss'],label='AdaGrad')
plt.plot(history_rmsprop.history['loss'],label='RMSProp')
plt.plot(history_nesterov.history['loss'],label='RMSProp+N')
plt.plot(history_adadelta.history['loss'],label='AdaDelta')
plt.plot(history_adam.history['loss'],label='Adam')
plt.title('Epoch vs training loss without dropout')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(loc=plt_loc)
plt.grid()
plt.show()
```



Part 3

Add dropout (probability 0.2 for input layer and 0.5 for hidden layers) and train the neural network again using all the five methods for 200 epochs. Compare the training loss with that in part 2. Which method performs the best? For the five methods, compare their training time (to finish 200 epochs with dropout) to the training time in part 2 (to finish 200 epochs without dropout). (5)

Code

Build model with dropout

In [4]:
```
from keras import Sequential
from keras.layers import Flatten, Dense, Dropout
activation_method='relu'
kregularizer_method='l2'
init_method = 'HeNormal'
def create_model():
    model2 = Sequential()
    model2.add(Flatten())
    model2.add(Dropout(0.2))
    model2.add(Dense(1000,activation=activation_method,kernel_regularizer=kregula
rizer_method,kernel_initializer=init_method))
    model2.add(Dropout(0.5))
    model2.add(Dense(1000,activation=activation_method,kernel_regularizer=kregula
rizer_method,kernel_initializer=init_method))
    model2.add(Dropout(0.5))
    model2.add(Dense(10, activation='softmax'))
    return model2
```

Adagrad

In [5]:
```
model_adagrad_dropout=create_model()
model_adagrad_dropout.compile(loss='categorical_crossentropy', optimizer='Adagra
d', metrics=['accuracy'])
history_adagrad_dropout = model_adagrad_dropout.fit(X_train, y_train, batch_size=
128, epochs=200, validation_data=(X_test,y_test),shuffle=True,verbose=2)
```

```
Epoch 1/200
391/391 - 7s - loss: 41.4371 - accuracy: 0.1482 - val_loss: 40.1760 - val_accura
cy: 0.2688
Epoch 2/200
391/391 - 6s - loss: 39.3439 - accuracy: 0.1916 - val_loss: 38.3034 - val_accura
cy: 0.2857
Epoch 3/200
391/391 - 6s - loss: 37.5195 - accuracy: 0.2138 - val_loss: 36.5496 - val_accura
cy: 0.2965
Epoch 4/200
391/391 - 6s - loss: 35.8131 - accuracy: 0.2311 - val_loss: 34.8890 - val_accura
cy: 0.3156
Epoch 5/200
391/391 - 6s - loss: 34.2016 - accuracy: 0.2466 - val_loss: 33.3311 - val_accura
cy: 0.3154
Epoch 6/200
391/391 - 6s - loss: 32.6777 - accuracy: 0.2587 - val_loss: 31.8498 - val_accura
cy: 0.3263
Epoch 7/200
391/391 - 6s - loss: 31.2375 - accuracy: 0.2666 - val_loss: 30.4499 - val_accura
cy: 0.3287
Epoch 8/200
391/391 - 6s - loss: 29.8705 - accuracy: 0.2750 - val_loss: 29.1196 - val_accura
cy: 0.3345
Epoch 9/200
391/391 - 6s - loss: 28.5775 - accuracy: 0.2764 - val_loss: 27.8547 - val_accura
cy: 0.3394
Epoch 10/200
391/391 - 6s - loss: 27.3447 - accuracy: 0.2891 - val_loss: 26.6635 - val_accura
cy: 0.3358
Epoch 11/200
391/391 - 6s - loss: 26.1764 - accuracy: 0.2950 - val_loss: 25.5213 - val_accura
cy: 0.3430
Epoch 12/200
391/391 - 6s - loss: 25.0673 - accuracy: 0.2974 - val_loss: 24.4402 - val_accura
cy: 0.3455
Epoch 13/200
391/391 - 6s - loss: 24.0079 - accuracy: 0.3038 - val_loss: 23.4113 - val_accura
cy: 0.3494
Epoch 14/200
391/391 - 6s - loss: 23.0068 - accuracy: 0.3059 - val_loss: 22.4370 - val_accura
cy: 0.3521
Epoch 15/200
391/391 - 6s - loss: 22.0546 - accuracy: 0.3104 - val_loss: 21.5064 - val_accura
cy: 0.3590
Epoch 16/200
391/391 - 6s - loss: 21.1438 - accuracy: 0.3148 - val_loss: 20.6201 - val_accura
cy: 0.3591
Epoch 17/200
391/391 - 6s - loss: 20.2845 - accuracy: 0.3171 - val_loss: 19.7770 - val_accura
cy: 0.3636
Epoch 18/200
391/391 - 6s - loss: 19.4571 - accuracy: 0.3219 - val_loss: 18.9748 - val_accura
cy: 0.3629
Epoch 19/200
391/391 - 6s - loss: 18.6774 - accuracy: 0.3239 - val_loss: 18.2153 - val_accura
cy: 0.3650
Epoch 20/200
391/391 - 6s - loss: 17.9295 - accuracy: 0.3257 - val_loss: 17.4815 - val_accura
```

```
cy: 0.3721
Epoch 21/200
391/391 - 6s - loss: 17.2193 - accuracy: 0.3296 - val_loss: 16.7899 - val_accura
cy: 0.3706
Epoch 22/200
391/391 - 6s - loss: 16.5405 - accuracy: 0.3324 - val_loss: 16.1343 - val_accura
cy: 0.3699
Epoch 23/200
391/391 - 6s - loss: 15.8965 - accuracy: 0.3347 - val_loss: 15.5019 - val_accura
cy: 0.3699
Epoch 24/200
391/391 - 6s - loss: 15.2816 - accuracy: 0.3381 - val_loss: 14.8994 - val_accura
cy: 0.3746
Epoch 25/200
391/391 - 6s - loss: 14.6953 - accuracy: 0.3390 - val_loss: 14.3246 - val_accura
cy: 0.3810
Epoch 26/200
391/391 - 6s - loss: 14.1346 - accuracy: 0.3418 - val_loss: 13.7867 - val_accura
cy: 0.3755
Epoch 27/200
391/391 - 6s - loss: 13.6036 - accuracy: 0.3444 - val_loss: 13.2605 - val_accura
cy: 0.3819
Epoch 28/200
391/391 - 6s - loss: 13.0900 - accuracy: 0.3467 - val_loss: 12.7660 - val_accura
cy: 0.3845
Epoch 29/200
391/391 - 6s - loss: 12.6055 - accuracy: 0.3487 - val_loss: 12.2882 - val_accura
cy: 0.3853
Epoch 30/200
391/391 - 6s - loss: 12.1400 - accuracy: 0.3513 - val_loss: 11.8368 - val_accura
cy: 0.3862
Epoch 31/200
391/391 - 6s - loss: 11.6987 - accuracy: 0.3544 - val_loss: 11.4061 - val_accura
cy: 0.3886
Epoch 32/200
391/391 - 6s - loss: 11.2776 - accuracy: 0.3531 - val_loss: 10.9905 - val_accura
cy: 0.3929
Epoch 33/200
391/391 - 6s - loss: 10.8737 - accuracy: 0.3558 - val_loss: 10.5943 - val_accura
cy: 0.3952
Epoch 34/200
391/391 - 6s - loss: 10.4881 - accuracy: 0.3590 - val_loss: 10.2213 - val_accura
cy: 0.3929
Epoch 35/200
391/391 - 6s - loss: 10.1190 - accuracy: 0.3610 - val_loss: 9.8605 - val_accurac
y: 0.3941
Epoch 36/200
391/391 - 6s - loss: 9.7717 - accuracy: 0.3595 - val_loss: 9.5164 - val_accurac
y: 0.3982
Epoch 37/200
391/391 - 6s - loss: 9.4353 - accuracy: 0.3599 - val_loss: 9.1939 - val_accurac
y: 0.3977
Epoch 38/200
391/391 - 6s - loss: 9.1163 - accuracy: 0.3638 - val_loss: 8.8796 - val_accurac
y: 0.3999
Epoch 39/200
391/391 - 6s - loss: 8.8079 - accuracy: 0.3644 - val_loss: 8.5809 - val_accurac
y: 0.3980
Epoch 40/200
```

```
391/391 - 6s - loss: 8.5167 - accuracy: 0.3665 - val_loss: 8.2926 - val_accurac
y: 0.4018
Epoch 41/200
391/391 - 6s - loss: 8.2352 - accuracy: 0.3697 - val_loss: 8.0207 - val_accurac
y: 0.4022
Epoch 42/200
391/391 - 6s - loss: 7.9684 - accuracy: 0.3711 - val_loss: 7.7569 - val_accurac
y: 0.4049
Epoch 43/200
391/391 - 5s - loss: 7.7125 - accuracy: 0.3717 - val_loss: 7.5086 - val_accurac
y: 0.4063
Epoch 44/200
391/391 - 6s - loss: 7.4692 - accuracy: 0.3706 - val_loss: 7.2695 - val_accurac
y: 0.4077
Epoch 45/200
391/391 - 6s - loss: 7.2348 - accuracy: 0.3736 - val_loss: 7.0433 - val_accurac
y: 0.4092
Epoch 46/200
391/391 - 6s - loss: 7.0090 - accuracy: 0.3755 - val_loss: 6.8241 - val_accurac
y: 0.4080
Epoch 47/200
391/391 - 6s - loss: 6.8002 - accuracy: 0.3740 - val_loss: 6.6184 - val_accurac
y: 0.4074
Epoch 48/200
391/391 - 6s - loss: 6.5931 - accuracy: 0.3766 - val_loss: 6.4157 - val_accurac
y: 0.4105
Epoch 49/200
391/391 - 6s - loss: 6.3949 - accuracy: 0.3792 - val_loss: 6.2230 - val_accurac
y: 0.4130
Epoch 50/200
391/391 - 6s - loss: 6.2106 - accuracy: 0.3779 - val_loss: 6.0390 - val_accurac
y: 0.4145
Epoch 51/200
391/391 - 6s - loss: 6.0315 - accuracy: 0.3819 - val_loss: 5.8694 - val_accurac
y: 0.4132
Epoch 52/200
391/391 - 6s - loss: 5.8600 - accuracy: 0.3813 - val_loss: 5.6996 - val_accurac
y: 0.4134
Epoch 53/200
391/391 - 6s - loss: 5.6936 - accuracy: 0.3847 - val_loss: 5.5423 - val_accurac
y: 0.4135
Epoch 54/200
391/391 - 6s - loss: 5.5377 - accuracy: 0.3849 - val_loss: 5.3883 - val_accurac
y: 0.4140
Epoch 55/200
391/391 - 6s - loss: 5.3873 - accuracy: 0.3879 - val_loss: 5.2391 - val_accurac
y: 0.4178
Epoch 56/200
391/391 - 6s - loss: 5.2458 - accuracy: 0.3877 - val_loss: 5.1000 - val_accurac
y: 0.4191
Epoch 57/200
391/391 - 6s - loss: 5.1074 - accuracy: 0.3850 - val_loss: 4.9632 - val_accurac
y: 0.4218
Epoch 58/200
391/391 - 6s - loss: 4.9748 - accuracy: 0.3870 - val_loss: 4.8356 - val_accurac
y: 0.4184
Epoch 59/200
391/391 - 6s - loss: 4.8501 - accuracy: 0.3867 - val_loss: 4.7143 - val_accurac
y: 0.4204
```

```
Epoch 60/200
391/391 - 6s - loss: 4.7263 - accuracy: 0.3927 - val_loss: 4.5957 - val_accurac
y: 0.4214
Epoch 61/200
391/391 - 6s - loss: 4.6123 - accuracy: 0.3911 - val_loss: 4.4845 - val_accurac
y: 0.4208
Epoch 62/200
391/391 - 6s - loss: 4.4995 - accuracy: 0.3936 - val_loss: 4.3709 - val_accurac
y: 0.4275
Epoch 63/200
391/391 - 6s - loss: 4.3927 - accuracy: 0.3943 - val_loss: 4.2696 - val_accurac
y: 0.4246
Epoch 64/200
391/391 - 6s - loss: 4.2913 - accuracy: 0.3935 - val_loss: 4.1696 - val_accurac
y: 0.4255
Epoch 65/200
391/391 - 6s - loss: 4.1927 - accuracy: 0.3963 - val_loss: 4.0785 - val_accurac
y: 0.4247
Epoch 66/200
391/391 - 6s - loss: 4.0991 - accuracy: 0.3973 - val_loss: 3.9884 - val_accurac
y: 0.4239
Epoch 67/200
391/391 - 6s - loss: 4.0111 - accuracy: 0.3958 - val_loss: 3.9007 - val_accurac
y: 0.4242
Epoch 68/200
391/391 - 6s - loss: 3.9219 - accuracy: 0.3973 - val_loss: 3.8146 - val_accurac
y: 0.4258
Epoch 69/200
391/391 - 6s - loss: 3.8409 - accuracy: 0.4000 - val_loss: 3.7312 - val_accurac
y: 0.4301
Epoch 70/200
391/391 - 6s - loss: 3.7606 - accuracy: 0.4002 - val_loss: 3.6515 - val_accurac
y: 0.4337
Epoch 71/200
391/391 - 6s - loss: 3.6848 - accuracy: 0.3995 - val_loss: 3.5828 - val_accurac
y: 0.4303
Epoch 72/200
391/391 - 6s - loss: 3.6104 - accuracy: 0.4032 - val_loss: 3.5125 - val_accurac
y: 0.4293
Epoch 73/200
391/391 - 6s - loss: 3.5412 - accuracy: 0.4028 - val_loss: 3.4396 - val_accurac
y: 0.4340
Epoch 74/200
391/391 - 6s - loss: 3.4763 - accuracy: 0.4012 - val_loss: 3.3751 - val_accurac
y: 0.4333
Epoch 75/200
391/391 - 6s - loss: 3.4074 - accuracy: 0.4042 - val_loss: 3.3135 - val_accurac
y: 0.4342
Epoch 76/200
391/391 - 6s - loss: 3.3465 - accuracy: 0.4052 - val_loss: 3.2546 - val_accurac
y: 0.4322
Epoch 77/200
391/391 - 6s - loss: 3.2875 - accuracy: 0.4078 - val_loss: 3.1937 - val_accurac
y: 0.4386
Epoch 78/200
391/391 - 6s - loss: 3.2308 - accuracy: 0.4051 - val_loss: 3.1391 - val_accurac
y: 0.4381
Epoch 79/200
391/391 - 6s - loss: 3.1761 - accuracy: 0.4066 - val_loss: 3.0857 - val_accurac
```

```
y: 0.4387
Epoch 80/200
391/391 - 6s - loss: 3.1238 - accuracy: 0.4080 - val_loss: 3.0371 - val_accurac
y: 0.4362
Epoch 81/200
391/391 - 6s - loss: 3.0729 - accuracy: 0.4096 - val_loss: 2.9861 - val_accurac
y: 0.4389
Epoch 82/200
391/391 - 6s - loss: 3.0250 - accuracy: 0.4075 - val_loss: 2.9391 - val_accurac
y: 0.4396
Epoch 83/200
391/391 - 6s - loss: 2.9789 - accuracy: 0.4130 - val_loss: 2.8920 - val_accurac
y: 0.4409
Epoch 84/200
391/391 - 6s - loss: 2.9315 - accuracy: 0.4106 - val_loss: 2.8492 - val_accurac
y: 0.4415
Epoch 85/200
391/391 - 6s - loss: 2.8897 - accuracy: 0.4130 - val_loss: 2.8095 - val_accurac
y: 0.4383
Epoch 86/200
391/391 - 6s - loss: 2.8488 - accuracy: 0.4133 - val_loss: 2.7673 - val_accurac
y: 0.4432
Epoch 87/200
391/391 - 6s - loss: 2.8136 - accuracy: 0.4098 - val_loss: 2.7317 - val_accurac
y: 0.4421
Epoch 88/200
391/391 - 6s - loss: 2.7701 - accuracy: 0.4155 - val_loss: 2.6927 - val_accurac
y: 0.4432
Epoch 89/200
391/391 - 6s - loss: 2.7338 - accuracy: 0.4162 - val_loss: 2.6555 - val_accurac
y: 0.4439
Epoch 90/200
391/391 - 6s - loss: 2.6984 - accuracy: 0.4160 - val_loss: 2.6212 - val_accurac
y: 0.4429
Epoch 91/200
391/391 - 6s - loss: 2.6651 - accuracy: 0.4160 - val_loss: 2.5878 - val_accurac
y: 0.4462
Epoch 92/200
391/391 - 6s - loss: 2.6344 - accuracy: 0.4141 - val_loss: 2.5590 - val_accurac
y: 0.4420
Epoch 93/200
391/391 - 6s - loss: 2.6044 - accuracy: 0.4166 - val_loss: 2.5303 - val_accurac
y: 0.4428
Epoch 94/200
391/391 - 6s - loss: 2.5715 - accuracy: 0.4167 - val_loss: 2.4993 - val_accurac
y: 0.4461
Epoch 95/200
391/391 - 6s - loss: 2.5396 - accuracy: 0.4180 - val_loss: 2.4702 - val_accurac
y: 0.4446
Epoch 96/200
391/391 - 6s - loss: 2.5132 - accuracy: 0.4183 - val_loss: 2.4432 - val_accurac
y: 0.4462
Epoch 97/200
391/391 - 6s - loss: 2.4853 - accuracy: 0.4198 - val_loss: 2.4166 - val_accurac
y: 0.4448
Epoch 98/200
391/391 - 6s - loss: 2.4606 - accuracy: 0.4210 - val_loss: 2.3909 - val_accurac
y: 0.4473
Epoch 99/200
```

```
391/391 - 6s - loss: 2.4380 - accuracy: 0.4210 - val_loss: 2.3660 - val_accurac
y: 0.4486
Epoch 100/200
391/391 - 6s - loss: 2.4114 - accuracy: 0.4208 - val_loss: 2.3430 - val_accurac
y: 0.4512
Epoch 101/200
391/391 - 6s - loss: 2.3890 - accuracy: 0.4226 - val_loss: 2.3225 - val_accurac
y: 0.4477
Epoch 102/200
391/391 - 6s - loss: 2.3680 - accuracy: 0.4222 - val_loss: 2.2994 - val_accurac
y: 0.4505
Epoch 103/200
391/391 - 6s - loss: 2.3464 - accuracy: 0.4226 - val_loss: 2.2789 - val_accurac
y: 0.4506
Epoch 104/200
391/391 - 6s - loss: 2.3240 - accuracy: 0.4225 - val_loss: 2.2612 - val_accurac
y: 0.4484
Epoch 105/200
391/391 - 6s - loss: 2.3047 - accuracy: 0.4233 - val_loss: 2.2385 - val_accurac
y: 0.4524
Epoch 106/200
391/391 - 6s - loss: 2.2852 - accuracy: 0.4260 - val_loss: 2.2232 - val_accurac
y: 0.4536
Epoch 107/200
391/391 - 6s - loss: 2.2670 - accuracy: 0.4259 - val_loss: 2.2057 - val_accurac
y: 0.4474
Epoch 108/200
391/391 - 6s - loss: 2.2494 - accuracy: 0.4265 - val_loss: 2.1866 - val_accurac
y: 0.4497
Epoch 109/200
391/391 - 6s - loss: 2.2302 - accuracy: 0.4295 - val_loss: 2.1697 - val_accurac
y: 0.4548
Epoch 110/200
391/391 - 6s - loss: 2.2161 - accuracy: 0.4257 - val_loss: 2.1526 - val_accurac
y: 0.4543
Epoch 111/200
391/391 - 6s - loss: 2.1989 - accuracy: 0.4274 - val_loss: 2.1382 - val_accurac
y: 0.4553
Epoch 112/200
391/391 - 6s - loss: 2.1823 - accuracy: 0.4304 - val_loss: 2.1225 - val_accurac
y: 0.4574
Epoch 113/200
391/391 - 6s - loss: 2.1690 - accuracy: 0.4275 - val_loss: 2.1086 - val_accurac
y: 0.4540
Epoch 114/200
391/391 - 6s - loss: 2.1540 - accuracy: 0.4265 - val_loss: 2.0945 - val_accurac
y: 0.4568
Epoch 115/200
391/391 - 6s - loss: 2.1403 - accuracy: 0.4301 - val_loss: 2.0803 - val_accurac
y: 0.4541
Epoch 116/200
391/391 - 6s - loss: 2.1269 - accuracy: 0.4306 - val_loss: 2.0680 - val_accurac
y: 0.4547
Epoch 117/200
391/391 - 6s - loss: 2.1120 - accuracy: 0.4306 - val_loss: 2.0562 - val_accurac
y: 0.4572
Epoch 118/200
391/391 - 6s - loss: 2.1026 - accuracy: 0.4327 - val_loss: 2.0450 - val_accurac
y: 0.4540
```

```
Epoch 119/200
391/391 - 6s - loss: 2.0885 - accuracy: 0.4302 - val_loss: 2.0317 - val_accurac
y: 0.4587
Epoch 120/200
391/391 - 6s - loss: 2.0744 - accuracy: 0.4356 - val_loss: 2.0220 - val_accurac
y: 0.4565
Epoch 121/200
391/391 - 6s - loss: 2.0660 - accuracy: 0.4304 - val_loss: 2.0078 - val_accurac
y: 0.4603
Epoch 122/200
391/391 - 6s - loss: 2.0571 - accuracy: 0.4324 - val_loss: 1.9990 - val_accurac
y: 0.4600
Epoch 123/200
391/391 - 6s - loss: 2.0476 - accuracy: 0.4356 - val_loss: 1.9913 - val_accurac
y: 0.4562
Epoch 124/200
391/391 - 6s - loss: 2.0332 - accuracy: 0.4357 - val_loss: 1.9804 - val_accurac
y: 0.4586
Epoch 125/200
391/391 - 6s - loss: 2.0252 - accuracy: 0.4345 - val_loss: 1.9690 - val_accurac
y: 0.4603
Epoch 126/200
391/391 - 6s - loss: 2.0158 - accuracy: 0.4344 - val_loss: 1.9600 - val_accurac
y: 0.4602
Epoch 127/200
391/391 - 6s - loss: 2.0063 - accuracy: 0.4363 - val_loss: 1.9531 - val_accurac
y: 0.4601
Epoch 128/200
391/391 - 6s - loss: 1.9961 - accuracy: 0.4374 - val_loss: 1.9440 - val_accurac
y: 0.4603
Epoch 129/200
391/391 - 6s - loss: 1.9907 - accuracy: 0.4334 - val_loss: 1.9342 - val_accurac
y: 0.4612
Epoch 130/200
391/391 - 6s - loss: 1.9807 - accuracy: 0.4345 - val_loss: 1.9305 - val_accurac
y: 0.4599
Epoch 131/200
391/391 - 6s - loss: 1.9744 - accuracy: 0.4375 - val_loss: 1.9191 - val_accurac
y: 0.4626
Epoch 132/200
391/391 - 6s - loss: 1.9657 - accuracy: 0.4375 - val_loss: 1.9131 - val_accurac
y: 0.4638
Epoch 133/200
391/391 - 6s - loss: 1.9557 - accuracy: 0.4372 - val_loss: 1.9058 - val_accurac
y: 0.4620
Epoch 134/200
391/391 - 6s - loss: 1.9499 - accuracy: 0.4397 - val_loss: 1.8979 - val_accurac
y: 0.4635
Epoch 135/200
391/391 - 6s - loss: 1.9442 - accuracy: 0.4370 - val_loss: 1.8900 - val_accurac
y: 0.4630
Epoch 136/200
391/391 - 6s - loss: 1.9340 - accuracy: 0.4406 - val_loss: 1.8839 - val_accurac
y: 0.4646
Epoch 137/200
391/391 - 6s - loss: 1.9300 - accuracy: 0.4408 - val_loss: 1.8800 - val_accurac
y: 0.4643
Epoch 138/200
391/391 - 6s - loss: 1.9241 - accuracy: 0.4370 - val_loss: 1.8708 - val_accurac
```

```
y: 0.4649
Epoch 139/200
391/391 - 6s - loss: 1.9178 - accuracy: 0.4397 - val_loss: 1.8673 - val_accurac
y: 0.4631
Epoch 140/200
391/391 - 6s - loss: 1.9113 - accuracy: 0.4410 - val_loss: 1.8616 - val_accurac
y: 0.4658
Epoch 141/200
391/391 - 6s - loss: 1.9058 - accuracy: 0.4414 - val_loss: 1.8573 - val_accurac
y: 0.4632
Epoch 142/200
391/391 - 6s - loss: 1.9002 - accuracy: 0.4404 - val_loss: 1.8511 - val_accurac
y: 0.4646
Epoch 143/200
391/391 - 6s - loss: 1.8940 - accuracy: 0.4417 - val_loss: 1.8444 - val_accurac
y: 0.4662
Epoch 144/200
391/391 - 6s - loss: 1.8916 - accuracy: 0.4427 - val_loss: 1.8378 - val_accurac
y: 0.4676
Epoch 145/200
391/391 - 6s - loss: 1.8828 - accuracy: 0.4421 - val_loss: 1.8348 - val_accurac
y: 0.4662
Epoch 146/200
391/391 - 6s - loss: 1.8785 - accuracy: 0.4440 - val_loss: 1.8287 - val_accurac
y: 0.4654
Epoch 147/200
391/391 - 6s - loss: 1.8728 - accuracy: 0.4425 - val_loss: 1.8244 - val_accurac
y: 0.4647
Epoch 148/200
391/391 - 6s - loss: 1.8704 - accuracy: 0.4446 - val_loss: 1.8201 - val_accurac
y: 0.4668
Epoch 149/200
391/391 - 6s - loss: 1.8659 - accuracy: 0.4437 - val_loss: 1.8156 - val_accurac
y: 0.4698
Epoch 150/200
391/391 - 6s - loss: 1.8598 - accuracy: 0.4424 - val_loss: 1.8112 - val_accurac
y: 0.4685
Epoch 151/200
391/391 - 6s - loss: 1.8548 - accuracy: 0.4464 - val_loss: 1.8080 - val_accurac
y: 0.4648
Epoch 152/200
391/391 - 6s - loss: 1.8494 - accuracy: 0.4462 - val_loss: 1.8029 - val_accurac
y: 0.4673
Epoch 153/200
391/391 - 6s - loss: 1.8496 - accuracy: 0.4441 - val_loss: 1.7987 - val_accurac
y: 0.4669
Epoch 154/200
391/391 - 6s - loss: 1.8432 - accuracy: 0.4450 - val_loss: 1.7978 - val_accurac
y: 0.4641
Epoch 155/200
391/391 - 6s - loss: 1.8383 - accuracy: 0.4455 - val_loss: 1.7911 - val_accurac
y: 0.4691
Epoch 156/200
391/391 - 6s - loss: 1.8368 - accuracy: 0.4459 - val_loss: 1.7880 - val_accurac
y: 0.4690
Epoch 157/200
391/391 - 6s - loss: 1.8331 - accuracy: 0.4469 - val_loss: 1.7837 - val_accurac
y: 0.4693
Epoch 158/200
```

```
391/391 - 6s - loss: 1.8284 - accuracy: 0.4455 - val_loss: 1.7817 - val_accurac
y: 0.4717
Epoch 159/200
391/391 - 6s - loss: 1.8261 - accuracy: 0.4445 - val_loss: 1.7778 - val_accurac
y: 0.4703
Epoch 160/200
391/391 - 6s - loss: 1.8219 - accuracy: 0.4477 - val_loss: 1.7740 - val_accurac
y: 0.4681
Epoch 161/200
391/391 - 6s - loss: 1.8196 - accuracy: 0.4460 - val_loss: 1.7709 - val_accurac
y: 0.4700
Epoch 162/200
391/391 - 6s - loss: 1.8153 - accuracy: 0.4464 - val_loss: 1.7678 - val_accurac
y: 0.4687
Epoch 163/200
391/391 - 6s - loss: 1.8127 - accuracy: 0.4490 - val_loss: 1.7647 - val_accurac
y: 0.4745
Epoch 164/200
391/391 - 6s - loss: 1.8109 - accuracy: 0.4479 - val_loss: 1.7641 - val_accurac
y: 0.4701
Epoch 165/200
391/391 - 6s - loss: 1.8084 - accuracy: 0.4469 - val_loss: 1.7649 - val_accurac
y: 0.4664
Epoch 166/200
391/391 - 6s - loss: 1.8045 - accuracy: 0.4478 - val_loss: 1.7558 - val_accurac
y: 0.4742
Epoch 167/200
391/391 - 6s - loss: 1.8010 - accuracy: 0.4491 - val_loss: 1.7560 - val_accurac
y: 0.4744
Epoch 168/200
391/391 - 6s - loss: 1.7963 - accuracy: 0.4511 - val_loss: 1.7541 - val_accurac
y: 0.4707
Epoch 169/200
391/391 - 6s - loss: 1.7949 - accuracy: 0.4528 - val_loss: 1.7480 - val_accurac
y: 0.4745
Epoch 170/200
391/391 - 6s - loss: 1.7895 - accuracy: 0.4508 - val_loss: 1.7474 - val_accurac
y: 0.4725
Epoch 171/200
391/391 - 6s - loss: 1.7904 - accuracy: 0.4501 - val_loss: 1.7448 - val_accurac
y: 0.4740
Epoch 172/200
391/391 - 6s - loss: 1.7872 - accuracy: 0.4515 - val_loss: 1.7415 - val_accurac
y: 0.4728
Epoch 173/200
391/391 - 6s - loss: 1.7838 - accuracy: 0.4517 - val_loss: 1.7406 - val_accurac
y: 0.4744
Epoch 174/200
391/391 - 6s - loss: 1.7812 - accuracy: 0.4516 - val_loss: 1.7364 - val_accurac
y: 0.4734
Epoch 175/200
391/391 - 6s - loss: 1.7787 - accuracy: 0.4506 - val_loss: 1.7349 - val_accurac
y: 0.4752
Epoch 176/200
391/391 - 6s - loss: 1.7763 - accuracy: 0.4521 - val_loss: 1.7375 - val_accurac
y: 0.4732
Epoch 177/200
391/391 - 6s - loss: 1.7776 - accuracy: 0.4508 - val_loss: 1.7323 - val_accurac
y: 0.4743
```

```
Epoch 178/200
391/391 - 6s - loss: 1.7750 - accuracy: 0.4515 - val_loss: 1.7305 - val_accurac
y: 0.4755
Epoch 179/200
391/391 - 6s - loss: 1.7714 - accuracy: 0.4518 - val_loss: 1.7286 - val_accurac
y: 0.4797
Epoch 180/200
391/391 - 6s - loss: 1.7688 - accuracy: 0.4539 - val_loss: 1.7245 - val_accurac
y: 0.4734
Epoch 181/200
391/391 - 6s - loss: 1.7667 - accuracy: 0.4534 - val_loss: 1.7221 - val_accurac
y: 0.4772
Epoch 182/200
391/391 - 6s - loss: 1.7670 - accuracy: 0.4517 - val_loss: 1.7208 - val_accurac
y: 0.4771
Epoch 183/200
391/391 - 6s - loss: 1.7625 - accuracy: 0.4539 - val_loss: 1.7210 - val_accurac
y: 0.4765
Epoch 184/200
391/391 - 6s - loss: 1.7607 - accuracy: 0.4551 - val_loss: 1.7176 - val_accurac
y: 0.4741
Epoch 185/200
391/391 - 6s - loss: 1.7588 - accuracy: 0.4563 - val_loss: 1.7157 - val_accurac
y: 0.4766
Epoch 186/200
391/391 - 6s - loss: 1.7592 - accuracy: 0.4527 - val_loss: 1.7143 - val_accurac
y: 0.4778
Epoch 187/200
391/391 - 6s - loss: 1.7577 - accuracy: 0.4542 - val_loss: 1.7209 - val_accurac
y: 0.4723
Epoch 188/200
391/391 - 6s - loss: 1.7568 - accuracy: 0.4539 - val_loss: 1.7119 - val_accurac
y: 0.4767
Epoch 189/200
391/391 - 6s - loss: 1.7532 - accuracy: 0.4548 - val_loss: 1.7090 - val_accurac
y: 0.4786
Epoch 190/200
391/391 - 6s - loss: 1.7535 - accuracy: 0.4537 - val_loss: 1.7067 - val_accurac
y: 0.4810
Epoch 191/200
391/391 - 6s - loss: 1.7490 - accuracy: 0.4557 - val_loss: 1.7079 - val_accurac
y: 0.4750
Epoch 192/200
391/391 - 6s - loss: 1.7485 - accuracy: 0.4560 - val_loss: 1.7057 - val_accurac
y: 0.4779
Epoch 193/200
391/391 - 6s - loss: 1.7464 - accuracy: 0.4546 - val_loss: 1.7029 - val_accurac
y: 0.4812
Epoch 194/200
391/391 - 6s - loss: 1.7447 - accuracy: 0.4567 - val_loss: 1.7024 - val_accurac
y: 0.4792
Epoch 195/200
391/391 - 6s - loss: 1.7430 - accuracy: 0.4567 - val_loss: 1.7019 - val_accurac
y: 0.4740
Epoch 196/200
391/391 - 6s - loss: 1.7409 - accuracy: 0.4589 - val_loss: 1.7003 - val_accurac
y: 0.4769
Epoch 197/200
391/391 - 6s - loss: 1.7416 - accuracy: 0.4566 - val_loss: 1.6975 - val_accurac
```

```
y: 0.4800
Epoch 198/200
391/391 - 6s - loss: 1.7400 - accuracy: 0.4559 - val_loss: 1.6989 - val_accurac
y: 0.4778
Epoch 199/200
391/391 - 6s - loss: 1.7371 - accuracy: 0.4573 - val_loss: 1.6970 - val_accurac
y: 0.4772
Epoch 200/200
391/391 - 6s - loss: 1.7369 - accuracy: 0.4575 - val_loss: 1.6934 - val_accurac
y: 0.4811
```

RMSProp

In [6]:
```
model_rmsprop_dropout=create_model()
model_rmsprop_dropout.compile(loss='categorical_crossentropy', optimizer='RMSprop', metrics=['accuracy'])
history_rmsprop_dropout = model_rmsprop_dropout.fit(X_train, y_train, batch_size=128, epochs=200, validation_data=(X_test,y_test),shuffle=True,verbose=2)
```

```
Epoch 1/200
391/391 - 8s - loss: 5.8783 - accuracy: 0.1698 - val_loss: 2.2167 - val_accurac
y: 0.2069
Epoch 2/200
391/391 - 7s - loss: 2.2653 - accuracy: 0.2124 - val_loss: 2.1549 - val_accurac
y: 0.2408
Epoch 3/200
391/391 - 7s - loss: 2.2435 - accuracy: 0.2188 - val_loss: 2.2449 - val_accurac
y: 0.2117
Epoch 4/200
391/391 - 7s - loss: 2.2158 - accuracy: 0.2260 - val_loss: 2.1279 - val_accurac
y: 0.2391
Epoch 5/200
391/391 - 7s - loss: 2.1949 - accuracy: 0.2335 - val_loss: 2.1902 - val_accurac
y: 0.2146
Epoch 6/200
391/391 - 7s - loss: 2.1704 - accuracy: 0.2409 - val_loss: 2.1071 - val_accurac
y: 0.2821
Epoch 7/200
391/391 - 7s - loss: 2.1513 - accuracy: 0.2411 - val_loss: 2.1132 - val_accurac
y: 0.3124
Epoch 8/200
391/391 - 7s - loss: 2.1384 - accuracy: 0.2440 - val_loss: 2.0829 - val_accurac
y: 0.2926
Epoch 9/200
391/391 - 7s - loss: 2.1316 - accuracy: 0.2488 - val_loss: 2.1523 - val_accurac
y: 0.2347
Epoch 10/200
391/391 - 7s - loss: 2.1255 - accuracy: 0.2488 - val_loss: 2.1247 - val_accurac
y: 0.2586
Epoch 11/200
391/391 - 7s - loss: 2.1248 - accuracy: 0.2474 - val_loss: 2.0588 - val_accurac
y: 0.3051
Epoch 12/200
391/391 - 7s - loss: 2.1267 - accuracy: 0.2497 - val_loss: 2.1534 - val_accurac
y: 0.2710
Epoch 13/200
391/391 - 7s - loss: 2.1249 - accuracy: 0.2484 - val_loss: 2.0783 - val_accurac
y: 0.3004
Epoch 14/200
391/391 - 7s - loss: 2.1198 - accuracy: 0.2510 - val_loss: 2.1119 - val_accurac
y: 0.2469
Epoch 15/200
391/391 - 8s - loss: 2.1202 - accuracy: 0.2472 - val_loss: 2.1107 - val_accurac
y: 0.2492
Epoch 16/200
391/391 - 8s - loss: 2.1240 - accuracy: 0.2498 - val_loss: 2.0981 - val_accurac
y: 0.2818
Epoch 17/200
391/391 - 8s - loss: 2.1236 - accuracy: 0.2485 - val_loss: 2.1338 - val_accurac
y: 0.2628
Epoch 18/200
391/391 - 8s - loss: 2.1239 - accuracy: 0.2496 - val_loss: 2.0977 - val_accurac
y: 0.3110
Epoch 19/200
391/391 - 8s - loss: 2.1191 - accuracy: 0.2496 - val_loss: 2.0980 - val_accurac
y: 0.3186
Epoch 20/200
391/391 - 8s - loss: 2.1173 - accuracy: 0.2523 - val_loss: 2.1075 - val_accurac
```

```
y: 0.2958
Epoch 21/200
391/391 - 8s - loss: 2.1220 - accuracy: 0.2490 - val_loss: 2.1070 - val_accurac
y: 0.2540
Epoch 22/200
391/391 - 8s - loss: 2.1243 - accuracy: 0.2451 - val_loss: 2.1057 - val_accurac
y: 0.2497
Epoch 23/200
391/391 - 8s - loss: 2.1213 - accuracy: 0.2464 - val_loss: 2.1215 - val_accurac
y: 0.2692
Epoch 24/200
391/391 - 8s - loss: 2.1215 - accuracy: 0.2476 - val_loss: 2.1364 - val_accurac
y: 0.2543
Epoch 25/200
391/391 - 8s - loss: 2.1226 - accuracy: 0.2491 - val_loss: 2.1067 - val_accurac
y: 0.2861
Epoch 26/200
391/391 - 8s - loss: 2.1200 - accuracy: 0.2490 - val_loss: 2.1377 - val_accurac
y: 0.2412
Epoch 27/200
391/391 - 8s - loss: 2.1213 - accuracy: 0.2503 - val_loss: 2.0894 - val_accurac
y: 0.2932
Epoch 28/200
391/391 - 8s - loss: 2.1176 - accuracy: 0.2512 - val_loss: 2.1061 - val_accurac
y: 0.2924
Epoch 29/200
391/391 - 8s - loss: 2.1210 - accuracy: 0.2498 - val_loss: 2.0693 - val_accurac
y: 0.3092
Epoch 30/200
391/391 - 8s - loss: 2.1234 - accuracy: 0.2457 - val_loss: 2.1391 - val_accurac
y: 0.2565
Epoch 31/200
391/391 - 8s - loss: 2.1222 - accuracy: 0.2473 - val_loss: 2.0977 - val_accurac
y: 0.2775
Epoch 32/200
391/391 - 8s - loss: 2.1226 - accuracy: 0.2487 - val_loss: 2.0471 - val_accurac
y: 0.3158
Epoch 33/200
391/391 - 8s - loss: 2.1187 - accuracy: 0.2509 - val_loss: 2.1226 - val_accurac
y: 0.2701
Epoch 34/200
391/391 - 8s - loss: 2.1219 - accuracy: 0.2476 - val_loss: 2.0900 - val_accurac
y: 0.3066
Epoch 35/200
391/391 - 8s - loss: 2.1227 - accuracy: 0.2484 - val_loss: 2.1174 - val_accurac
y: 0.2183
Epoch 36/200
391/391 - 8s - loss: 2.1211 - accuracy: 0.2470 - val_loss: 2.1390 - val_accurac
y: 0.2310
Epoch 37/200
391/391 - 8s - loss: 2.1236 - accuracy: 0.2441 - val_loss: 2.1210 - val_accurac
y: 0.2996
Epoch 38/200
391/391 - 8s - loss: 2.1216 - accuracy: 0.2474 - val_loss: 2.1080 - val_accurac
y: 0.2890
Epoch 39/200
391/391 - 8s - loss: 2.1217 - accuracy: 0.2463 - val_loss: 2.1136 - val_accurac
y: 0.2566
Epoch 40/200
```

```
391/391 - 8s - loss: 2.1237 - accuracy: 0.2451 - val_loss: 2.1174 - val_accurac
y: 0.2483
Epoch 41/200
391/391 - 8s - loss: 2.1194 - accuracy: 0.2490 - val_loss: 2.1026 - val_accurac
y: 0.2804
Epoch 42/200
391/391 - 8s - loss: 2.1206 - accuracy: 0.2456 - val_loss: 2.1223 - val_accurac
y: 0.2829
Epoch 43/200
391/391 - 8s - loss: 2.1224 - accuracy: 0.2448 - val_loss: 2.0993 - val_accurac
y: 0.2643
Epoch 44/200
391/391 - 8s - loss: 2.1210 - accuracy: 0.2475 - val_loss: 2.1053 - val_accurac
y: 0.2920
Epoch 45/200
391/391 - 8s - loss: 2.1231 - accuracy: 0.2440 - val_loss: 2.1043 - val_accurac
y: 0.2896
Epoch 46/200
391/391 - 7s - loss: 2.1186 - accuracy: 0.2506 - val_loss: 2.0923 - val_accurac
y: 0.2708
Epoch 47/200
391/391 - 7s - loss: 2.1208 - accuracy: 0.2467 - val_loss: 2.1223 - val_accurac
y: 0.2642
Epoch 48/200
391/391 - 7s - loss: 2.1209 - accuracy: 0.2453 - val_loss: 2.0838 - val_accurac
y: 0.2638
Epoch 49/200
391/391 - 7s - loss: 2.1252 - accuracy: 0.2447 - val_loss: 2.1223 - val_accurac
y: 0.2684
Epoch 50/200
391/391 - 7s - loss: 2.1175 - accuracy: 0.2477 - val_loss: 2.1099 - val_accurac
y: 0.2906
Epoch 51/200
391/391 - 7s - loss: 2.1208 - accuracy: 0.2454 - val_loss: 2.1247 - val_accurac
y: 0.2606
Epoch 52/200
391/391 - 7s - loss: 2.1201 - accuracy: 0.2494 - val_loss: 2.0784 - val_accurac
y: 0.2688
Epoch 53/200
391/391 - 8s - loss: 2.1226 - accuracy: 0.2449 - val_loss: 2.1084 - val_accurac
y: 0.2587
Epoch 54/200
391/391 - 8s - loss: 2.1238 - accuracy: 0.2429 - val_loss: 2.1390 - val_accurac
y: 0.2410
Epoch 55/200
391/391 - 8s - loss: 2.1208 - accuracy: 0.2460 - val_loss: 2.0809 - val_accurac
y: 0.2668
Epoch 56/200
391/391 - 8s - loss: 2.1204 - accuracy: 0.2421 - val_loss: 2.0724 - val_accurac
y: 0.2981
Epoch 57/200
391/391 - 8s - loss: 2.1213 - accuracy: 0.2467 - val_loss: 2.0534 - val_accurac
y: 0.2924
Epoch 58/200
391/391 - 8s - loss: 2.1217 - accuracy: 0.2449 - val_loss: 2.0912 - val_accurac
y: 0.3009
Epoch 59/200
391/391 - 7s - loss: 2.1199 - accuracy: 0.2449 - val_loss: 2.1445 - val_accurac
y: 0.2647
```

```
Epoch 60/200
391/391 - 7s - loss: 2.1212 - accuracy: 0.2446 - val_loss: 2.0940 - val_accurac
y: 0.2812
Epoch 61/200
391/391 - 7s - loss: 2.1203 - accuracy: 0.2455 - val_loss: 2.0864 - val_accurac
y: 0.2936
Epoch 62/200
391/391 - 8s - loss: 2.1255 - accuracy: 0.2433 - val_loss: 2.0896 - val_accurac
y: 0.3031
Epoch 63/200
391/391 - 8s - loss: 2.1156 - accuracy: 0.2464 - val_loss: 2.0940 - val_accurac
y: 0.2777
Epoch 64/200
391/391 - 8s - loss: 2.1255 - accuracy: 0.2426 - val_loss: 2.1128 - val_accurac
y: 0.2710
Epoch 65/200
391/391 - 8s - loss: 2.1195 - accuracy: 0.2464 - val_loss: 2.0967 - val_accurac
y: 0.2941
Epoch 66/200
391/391 - 8s - loss: 2.1173 - accuracy: 0.2468 - val_loss: 2.0996 - val_accurac
y: 0.2520
Epoch 67/200
391/391 - 8s - loss: 2.1221 - accuracy: 0.2440 - val_loss: 2.0714 - val_accurac
y: 0.2944
Epoch 68/200
391/391 - 8s - loss: 2.1217 - accuracy: 0.2451 - val_loss: 2.0683 - val_accurac
y: 0.2904
Epoch 69/200
391/391 - 8s - loss: 2.1224 - accuracy: 0.2434 - val_loss: 2.0831 - val_accurac
y: 0.2816
Epoch 70/200
391/391 - 8s - loss: 2.1203 - accuracy: 0.2444 - val_loss: 2.1135 - val_accurac
y: 0.2758
Epoch 71/200
391/391 - 8s - loss: 2.1251 - accuracy: 0.2450 - val_loss: 2.0524 - val_accurac
y: 0.2932
Epoch 72/200
391/391 - 8s - loss: 2.1221 - accuracy: 0.2416 - val_loss: 2.0843 - val_accurac
y: 0.2944
Epoch 73/200
391/391 - 8s - loss: 2.1219 - accuracy: 0.2477 - val_loss: 2.0789 - val_accurac
y: 0.2985
Epoch 74/200
391/391 - 8s - loss: 2.1247 - accuracy: 0.2433 - val_loss: 2.0740 - val_accurac
y: 0.2368
Epoch 75/200
391/391 - 8s - loss: 2.1247 - accuracy: 0.2400 - val_loss: 2.1196 - val_accurac
y: 0.2794
Epoch 76/200
391/391 - 8s - loss: 2.1192 - accuracy: 0.2448 - val_loss: 2.1250 - val_accurac
y: 0.2901
Epoch 77/200
391/391 - 8s - loss: 2.1270 - accuracy: 0.2399 - val_loss: 2.1074 - val_accurac
y: 0.2112
Epoch 78/200
391/391 - 8s - loss: 2.1228 - accuracy: 0.2407 - val_loss: 2.1134 - val_accurac
y: 0.2491
Epoch 79/200
391/391 - 8s - loss: 2.1190 - accuracy: 0.2432 - val_loss: 2.0564 - val_accurac
```

```
y: 0.3135
Epoch 80/200
391/391 - 8s - loss: 2.1211 - accuracy: 0.2416 - val_loss: 2.0923 - val_accurac
y: 0.3042
Epoch 81/200
391/391 - 8s - loss: 2.1252 - accuracy: 0.2392 - val_loss: 2.0982 - val_accurac
y: 0.2755
Epoch 82/200
391/391 - 8s - loss: 2.1182 - accuracy: 0.2447 - val_loss: 2.0548 - val_accurac
y: 0.3067
Epoch 83/200
391/391 - 8s - loss: 2.1186 - accuracy: 0.2460 - val_loss: 2.0708 - val_accurac
y: 0.3030
Epoch 84/200
391/391 - 8s - loss: 2.1206 - accuracy: 0.2442 - val_loss: 2.1057 - val_accurac
y: 0.2762
Epoch 85/200
391/391 - 8s - loss: 2.1200 - accuracy: 0.2423 - val_loss: 2.0973 - val_accurac
y: 0.2915
Epoch 86/200
391/391 - 8s - loss: 2.1221 - accuracy: 0.2450 - val_loss: 2.1535 - val_accurac
y: 0.2305
Epoch 87/200
391/391 - 8s - loss: 2.1243 - accuracy: 0.2398 - val_loss: 2.0953 - val_accurac
y: 0.2537
Epoch 88/200
391/391 - 8s - loss: 2.1212 - accuracy: 0.2446 - val_loss: 2.0803 - val_accurac
y: 0.3135
Epoch 89/200
391/391 - 7s - loss: 2.1255 - accuracy: 0.2425 - val_loss: 2.0893 - val_accurac
y: 0.3021
Epoch 90/200
391/391 - 8s - loss: 2.1163 - accuracy: 0.2490 - val_loss: 2.1157 - val_accurac
y: 0.2394
Epoch 91/200
391/391 - 8s - loss: 2.1212 - accuracy: 0.2448 - val_loss: 2.1299 - val_accurac
y: 0.2684
Epoch 92/200
391/391 - 8s - loss: 2.1272 - accuracy: 0.2397 - val_loss: 2.1003 - val_accurac
y: 0.3025
Epoch 93/200
391/391 - 8s - loss: 2.1223 - accuracy: 0.2429 - val_loss: 2.1463 - val_accurac
y: 0.2495
Epoch 94/200
391/391 - 8s - loss: 2.1236 - accuracy: 0.2410 - val_loss: 2.1074 - val_accurac
y: 0.2402
Epoch 95/200
391/391 - 8s - loss: 2.1171 - accuracy: 0.2457 - val_loss: 2.1196 - val_accurac
y: 0.2657
Epoch 96/200
391/391 - 8s - loss: 2.1204 - accuracy: 0.2438 - val_loss: 2.1324 - val_accurac
y: 0.2436
Epoch 97/200
391/391 - 8s - loss: 2.1257 - accuracy: 0.2397 - val_loss: 2.0889 - val_accurac
y: 0.2544
Epoch 98/200
391/391 - 8s - loss: 2.1197 - accuracy: 0.2454 - val_loss: 2.0661 - val_accurac
y: 0.2624
Epoch 99/200
```

```
391/391 - 8s - loss: 2.1204 - accuracy: 0.2415 - val_loss: 2.1060 - val_accurac
y: 0.2925
Epoch 100/200
391/391 - 8s - loss: 2.1199 - accuracy: 0.2424 - val_loss: 2.0623 - val_accurac
y: 0.2673
Epoch 101/200
391/391 - 8s - loss: 2.1243 - accuracy: 0.2416 - val_loss: 2.0850 - val_accurac
y: 0.3004
Epoch 102/200
391/391 - 8s - loss: 2.1197 - accuracy: 0.2464 - val_loss: 2.0912 - val_accurac
y: 0.3132
Epoch 103/200
391/391 - 8s - loss: 2.1178 - accuracy: 0.2458 - val_loss: 2.0594 - val_accurac
y: 0.2818
Epoch 104/200
391/391 - 8s - loss: 2.1229 - accuracy: 0.2426 - val_loss: 2.0373 - val_accurac
y: 0.3137
Epoch 105/200
391/391 - 7s - loss: 2.1234 - accuracy: 0.2406 - val_loss: 2.1463 - val_accurac
y: 0.2135
Epoch 106/200
391/391 - 7s - loss: 2.1202 - accuracy: 0.2430 - val_loss: 2.0955 - val_accurac
y: 0.2552
Epoch 107/200
391/391 - 8s - loss: 2.1250 - accuracy: 0.2407 - val_loss: 2.0991 - val_accurac
y: 0.2466
Epoch 108/200
391/391 - 8s - loss: 2.1239 - accuracy: 0.2403 - val_loss: 2.1238 - val_accurac
y: 0.2548
Epoch 109/200
391/391 - 8s - loss: 2.1250 - accuracy: 0.2414 - val_loss: 2.0939 - val_accurac
y: 0.2539
Epoch 110/200
391/391 - 8s - loss: 2.1195 - accuracy: 0.2428 - val_loss: 2.0917 - val_accurac
y: 0.2809
Epoch 111/200
391/391 - 8s - loss: 2.1207 - accuracy: 0.2434 - val_loss: 2.0709 - val_accurac
y: 0.2819
Epoch 112/200
391/391 - 8s - loss: 2.1245 - accuracy: 0.2427 - val_loss: 2.0991 - val_accurac
y: 0.2364
Epoch 113/200
391/391 - 8s - loss: 2.1244 - accuracy: 0.2430 - val_loss: 2.0772 - val_accurac
y: 0.2857
Epoch 114/200
391/391 - 8s - loss: 2.1261 - accuracy: 0.2404 - val_loss: 2.0829 - val_accurac
y: 0.2794
Epoch 115/200
391/391 - 8s - loss: 2.1206 - accuracy: 0.2442 - val_loss: 2.1022 - val_accurac
y: 0.2627
Epoch 116/200
391/391 - 8s - loss: 2.1262 - accuracy: 0.2423 - val_loss: 2.0861 - val_accurac
y: 0.2844
Epoch 117/200
391/391 - 8s - loss: 2.1195 - accuracy: 0.2428 - val_loss: 2.1156 - val_accurac
y: 0.2424
Epoch 118/200
391/391 - 8s - loss: 2.1244 - accuracy: 0.2400 - val_loss: 2.0667 - val_accurac
y: 0.2606
```

```
Epoch 119/200
391/391 - 8s - loss: 2.1204 - accuracy: 0.2418 - val_loss: 2.1440 - val_accurac
y: 0.2431
Epoch 120/200
391/391 - 8s - loss: 2.1221 - accuracy: 0.2437 - val_loss: 2.0597 - val_accurac
y: 0.2771
Epoch 121/200
391/391 - 8s - loss: 2.1212 - accuracy: 0.2408 - val_loss: 2.0674 - val_accurac
y: 0.2805
Epoch 122/200
391/391 - 8s - loss: 2.1256 - accuracy: 0.2440 - val_loss: 2.1583 - val_accurac
y: 0.2024
Epoch 123/200
391/391 - 7s - loss: 2.1209 - accuracy: 0.2403 - val_loss: 2.0988 - val_accurac
y: 0.2514
Epoch 124/200
391/391 - 8s - loss: 2.1211 - accuracy: 0.2430 - val_loss: 2.0568 - val_accurac
y: 0.3158
Epoch 125/200
391/391 - 8s - loss: 2.1211 - accuracy: 0.2422 - val_loss: 2.0821 - val_accurac
y: 0.2660
Epoch 126/200
391/391 - 7s - loss: 2.1228 - accuracy: 0.2457 - val_loss: 2.0911 - val_accurac
y: 0.2961
Epoch 127/200
391/391 - 8s - loss: 2.1270 - accuracy: 0.2404 - val_loss: 2.1013 - val_accurac
y: 0.2819
Epoch 128/200
391/391 - 8s - loss: 2.1203 - accuracy: 0.2426 - val_loss: 2.0664 - val_accurac
y: 0.3008
Epoch 129/200
391/391 - 8s - loss: 2.1244 - accuracy: 0.2417 - val_loss: 2.0890 - val_accurac
y: 0.2406
Epoch 130/200
391/391 - 8s - loss: 2.1184 - accuracy: 0.2452 - val_loss: 2.0500 - val_accurac
y: 0.2843
Epoch 131/200
391/391 - 8s - loss: 2.1275 - accuracy: 0.2404 - val_loss: 2.0632 - val_accurac
y: 0.2871
Epoch 132/200
391/391 - 8s - loss: 2.1229 - accuracy: 0.2417 - val_loss: 2.0609 - val_accurac
y: 0.2702
Epoch 133/200
391/391 - 8s - loss: 2.1217 - accuracy: 0.2427 - val_loss: 2.0708 - val_accurac
y: 0.2938
Epoch 134/200
391/391 - 8s - loss: 2.1196 - accuracy: 0.2437 - val_loss: 2.0857 - val_accurac
y: 0.2530
Epoch 135/200
391/391 - 8s - loss: 2.1223 - accuracy: 0.2413 - val_loss: 2.0810 - val_accurac
y: 0.2538
Epoch 136/200
391/391 - 8s - loss: 2.1203 - accuracy: 0.2423 - val_loss: 2.0671 - val_accurac
y: 0.2924
Epoch 137/200
391/391 - 8s - loss: 2.1206 - accuracy: 0.2431 - val_loss: 2.0839 - val_accurac
y: 0.2503
Epoch 138/200
391/391 - 8s - loss: 2.1215 - accuracy: 0.2414 - val_loss: 2.0661 - val_accurac
```

```
y: 0.2926
Epoch 139/200
391/391 - 8s - loss: 2.1224 - accuracy: 0.2404 - val_loss: 2.1019 - val_accurac
y: 0.2900
Epoch 140/200
391/391 - 8s - loss: 2.1190 - accuracy: 0.2474 - val_loss: 2.1153 - val_accurac
y: 0.2348
Epoch 141/200
391/391 - 8s - loss: 2.1234 - accuracy: 0.2427 - val_loss: 2.0799 - val_accurac
y: 0.2918
Epoch 142/200
391/391 - 8s - loss: 2.1216 - accuracy: 0.2412 - val_loss: 2.0758 - val_accurac
y: 0.2549
Epoch 143/200
391/391 - 8s - loss: 2.1195 - accuracy: 0.2429 - val_loss: 2.0745 - val_accurac
y: 0.2973
Epoch 144/200
391/391 - 8s - loss: 2.1197 - accuracy: 0.2450 - val_loss: 2.1406 - val_accurac
y: 0.2175
Epoch 145/200
391/391 - 8s - loss: 2.1211 - accuracy: 0.2454 - val_loss: 2.0621 - val_accurac
y: 0.2755
Epoch 146/200
391/391 - 8s - loss: 2.1195 - accuracy: 0.2468 - val_loss: 2.0848 - val_accurac
y: 0.2642
Epoch 147/200
391/391 - 8s - loss: 2.1158 - accuracy: 0.2444 - val_loss: 2.0930 - val_accurac
y: 0.2874
Epoch 148/200
391/391 - 8s - loss: 2.1277 - accuracy: 0.2424 - val_loss: 2.0985 - val_accurac
y: 0.2551
Epoch 149/200
391/391 - 8s - loss: 2.1237 - accuracy: 0.2419 - val_loss: 2.0644 - val_accurac
y: 0.2874
Epoch 150/200
391/391 - 8s - loss: 2.1220 - accuracy: 0.2445 - val_loss: 2.0620 - val_accurac
y: 0.3090
Epoch 151/200
391/391 - 7s - loss: 2.1213 - accuracy: 0.2424 - val_loss: 2.0462 - val_accurac
y: 0.2775
Epoch 152/200
391/391 - 7s - loss: 2.1181 - accuracy: 0.2425 - val_loss: 2.0559 - val_accurac
y: 0.2835
Epoch 153/200
391/391 - 8s - loss: 2.1214 - accuracy: 0.2415 - val_loss: 2.1021 - val_accurac
y: 0.2770
Epoch 154/200
391/391 - 8s - loss: 2.1225 - accuracy: 0.2422 - val_loss: 2.1188 - val_accurac
y: 0.2744
Epoch 155/200
391/391 - 8s - loss: 2.1212 - accuracy: 0.2408 - val_loss: 2.0691 - val_accurac
y: 0.2951
Epoch 156/200
391/391 - 8s - loss: 2.1263 - accuracy: 0.2369 - val_loss: 2.0863 - val_accurac
y: 0.2532
Epoch 157/200
391/391 - 8s - loss: 2.1200 - accuracy: 0.2439 - val_loss: 2.0723 - val_accurac
y: 0.2917
Epoch 158/200
```

```
391/391 - 8s - loss: 2.1222 - accuracy: 0.2418 - val_loss: 2.0757 - val_accurac
y: 0.2642
Epoch 159/200
391/391 - 8s - loss: 2.1212 - accuracy: 0.2400 - val_loss: 2.1018 - val_accurac
y: 0.2817
Epoch 160/200
391/391 - 8s - loss: 2.1209 - accuracy: 0.2402 - val_loss: 2.1034 - val_accurac
y: 0.2874
Epoch 161/200
391/391 - 8s - loss: 2.1187 - accuracy: 0.2455 - val_loss: 2.0594 - val_accurac
y: 0.2807
Epoch 162/200
391/391 - 7s - loss: 2.1258 - accuracy: 0.2416 - val_loss: 2.0744 - val_accurac
y: 0.2508
Epoch 163/200
391/391 - 8s - loss: 2.1204 - accuracy: 0.2397 - val_loss: 2.1415 - val_accurac
y: 0.2656
Epoch 164/200
391/391 - 8s - loss: 2.1257 - accuracy: 0.2417 - val_loss: 2.0867 - val_accurac
y: 0.2772
Epoch 165/200
391/391 - 8s - loss: 2.1164 - accuracy: 0.2438 - val_loss: 2.0513 - val_accurac
y: 0.2938
Epoch 166/200
391/391 - 8s - loss: 2.1235 - accuracy: 0.2411 - val_loss: 2.1266 - val_accurac
y: 0.2592
Epoch 167/200
391/391 - 8s - loss: 2.1199 - accuracy: 0.2426 - val_loss: 2.0986 - val_accurac
y: 0.2901
Epoch 168/200
391/391 - 8s - loss: 2.1228 - accuracy: 0.2408 - val_loss: 2.0549 - val_accurac
y: 0.2879
Epoch 169/200
391/391 - 8s - loss: 2.1237 - accuracy: 0.2415 - val_loss: 2.0825 - val_accurac
y: 0.2747
Epoch 170/200
391/391 - 8s - loss: 2.1196 - accuracy: 0.2412 - val_loss: 2.0342 - val_accurac
y: 0.3107
Epoch 171/200
391/391 - 8s - loss: 2.1278 - accuracy: 0.2397 - val_loss: 2.0760 - val_accurac
y: 0.2934
Epoch 172/200
391/391 - 8s - loss: 2.1233 - accuracy: 0.2407 - val_loss: 2.0788 - val_accurac
y: 0.2883
Epoch 173/200
391/391 - 8s - loss: 2.1210 - accuracy: 0.2450 - val_loss: 2.0848 - val_accurac
y: 0.2826
Epoch 174/200
391/391 - 8s - loss: 2.1239 - accuracy: 0.2398 - val_loss: 2.0677 - val_accurac
y: 0.2860
Epoch 175/200
391/391 - 8s - loss: 2.1232 - accuracy: 0.2395 - val_loss: 2.0843 - val_accurac
y: 0.2916
Epoch 176/200
391/391 - 8s - loss: 2.1228 - accuracy: 0.2402 - val_loss: 2.0744 - val_accurac
y: 0.2912
Epoch 177/200
391/391 - 8s - loss: 2.1244 - accuracy: 0.2421 - val_loss: 2.0831 - val_accurac
y: 0.2731
```

```
Epoch 178/200
391/391 - 8s - loss: 2.1261 - accuracy: 0.2371 - val_loss: 2.0660 - val_accurac
y: 0.2969
Epoch 179/200
391/391 - 8s - loss: 2.1249 - accuracy: 0.2398 - val_loss: 2.0807 - val_accurac
y: 0.3024
Epoch 180/200
391/391 - 8s - loss: 2.1199 - accuracy: 0.2416 - val_loss: 2.0943 - val_accurac
y: 0.2807
Epoch 181/200
391/391 - 8s - loss: 2.1206 - accuracy: 0.2417 - val_loss: 2.1721 - val_accurac
y: 0.2500
Epoch 182/200
391/391 - 8s - loss: 2.1234 - accuracy: 0.2416 - val_loss: 2.0708 - val_accurac
y: 0.2673
Epoch 183/200
391/391 - 8s - loss: 2.1241 - accuracy: 0.2440 - val_loss: 2.1029 - val_accurac
y: 0.2669
Epoch 184/200
391/391 - 8s - loss: 2.1226 - accuracy: 0.2395 - val_loss: 2.0894 - val_accurac
y: 0.2493
Epoch 185/200
391/391 - 8s - loss: 2.1215 - accuracy: 0.2412 - val_loss: 2.0831 - val_accurac
y: 0.2844
Epoch 186/200
391/391 - 8s - loss: 2.1211 - accuracy: 0.2423 - val_loss: 2.0480 - val_accurac
y: 0.2678
Epoch 187/200
391/391 - 8s - loss: 2.1211 - accuracy: 0.2421 - val_loss: 2.0667 - val_accurac
y: 0.2840
Epoch 188/200
391/391 - 8s - loss: 2.1225 - accuracy: 0.2444 - val_loss: 2.0562 - val_accurac
y: 0.2966
Epoch 189/200
391/391 - 8s - loss: 2.1214 - accuracy: 0.2436 - val_loss: 2.0955 - val_accurac
y: 0.2465
Epoch 190/200
391/391 - 8s - loss: 2.1213 - accuracy: 0.2413 - val_loss: 2.1144 - val_accurac
y: 0.2539
Epoch 191/200
391/391 - 8s - loss: 2.1256 - accuracy: 0.2418 - val_loss: 2.0755 - val_accurac
y: 0.2689
Epoch 192/200
391/391 - 8s - loss: 2.1211 - accuracy: 0.2423 - val_loss: 2.1005 - val_accurac
y: 0.2701
Epoch 193/200
391/391 - 8s - loss: 2.1210 - accuracy: 0.2431 - val_loss: 2.1476 - val_accurac
y: 0.2432
Epoch 194/200
391/391 - 8s - loss: 2.1237 - accuracy: 0.2392 - val_loss: 2.0371 - val_accurac
y: 0.2959
Epoch 195/200
391/391 - 8s - loss: 2.1218 - accuracy: 0.2397 - val_loss: 2.0965 - val_accurac
y: 0.2401
Epoch 196/200
391/391 - 8s - loss: 2.1207 - accuracy: 0.2399 - val_loss: 2.1127 - val_accurac
y: 0.2200
Epoch 197/200
391/391 - 7s - loss: 2.1255 - accuracy: 0.2393 - val_loss: 2.1587 - val_accurac
```

```
y: 0.1917
Epoch 198/200
391/391 - 8s - loss: 2.1241 - accuracy: 0.2403 - val_loss: 2.0885 - val_accurac
y: 0.2791
Epoch 199/200
391/391 - 8s - loss: 2.1221 - accuracy: 0.2438 - val_loss: 2.0716 - val_accurac
y: 0.2554
Epoch 200/200
391/391 - 8s - loss: 2.1253 - accuracy: 0.2362 - val_loss: 2.0576 - val_accurac
y: 0.2864
```

RMSProp + Nesterov

In [7]:
```
model_nesterov_dropout=create_model()
model_nesterov_dropout.compile(loss='categorical_crossentropy', optimizer='Nadam'
, metrics=['accuracy'])
history_nesterov_dropout = model_nesterov_dropout.fit(X_train, y_train, batch_siz
e=128, epochs=200, validation_data=(X_test,y_test),shuffle=True,verbose=2)
```

```
Epoch 1/200
391/391 - 11s - loss: 7.3066 - accuracy: 0.2072 - val_loss: 2.1672 - val_accurac
y: 0.2793
Epoch 2/200
391/391 - 10s - loss: 2.1533 - accuracy: 0.2347 - val_loss: 2.0583 - val_accurac
y: 0.2644
Epoch 3/200
391/391 - 10s - loss: 2.1037 - accuracy: 0.2438 - val_loss: 2.0391 - val_accurac
y: 0.3030
Epoch 4/200
391/391 - 9s - loss: 2.0866 - accuracy: 0.2536 - val_loss: 2.0364 - val_accurac
y: 0.2912
Epoch 5/200
391/391 - 10s - loss: 2.0934 - accuracy: 0.2447 - val_loss: 2.0061 - val_accurac
y: 0.2974
Epoch 6/200
391/391 - 10s - loss: 2.0884 - accuracy: 0.2479 - val_loss: 2.0411 - val_accurac
y: 0.2944
Epoch 7/200
391/391 - 10s - loss: 2.0830 - accuracy: 0.2513 - val_loss: 2.0438 - val_accurac
y: 0.2762
Epoch 8/200
391/391 - 10s - loss: 2.0777 - accuracy: 0.2519 - val_loss: 1.9980 - val_accurac
y: 0.3296
Epoch 9/200
391/391 - 10s - loss: 2.0604 - accuracy: 0.2574 - val_loss: 2.0198 - val_accurac
y: 0.2981
Epoch 10/200
391/391 - 10s - loss: 2.0729 - accuracy: 0.2514 - val_loss: 2.0139 - val_accurac
y: 0.2897
Epoch 11/200
391/391 - 10s - loss: 2.0754 - accuracy: 0.2499 - val_loss: 2.0108 - val_accurac
y: 0.3032
Epoch 12/200
391/391 - 10s - loss: 2.0639 - accuracy: 0.2542 - val_loss: 2.0308 - val_accurac
y: 0.2517
Epoch 13/200
391/391 - 10s - loss: 2.0598 - accuracy: 0.2569 - val_loss: 2.0368 - val_accurac
y: 0.2757
Epoch 14/200
391/391 - 10s - loss: 2.0525 - accuracy: 0.2586 - val_loss: 2.0483 - val_accurac
y: 0.2620
Epoch 15/200
391/391 - 10s - loss: 2.0517 - accuracy: 0.2565 - val_loss: 2.0499 - val_accurac
y: 0.2943
Epoch 16/200
391/391 - 10s - loss: 2.0597 - accuracy: 0.2538 - val_loss: 2.0541 - val_accurac
y: 0.2911
Epoch 17/200
391/391 - 10s - loss: 2.0503 - accuracy: 0.2616 - val_loss: 2.0266 - val_accurac
y: 0.3091
Epoch 18/200
391/391 - 10s - loss: 2.0508 - accuracy: 0.2611 - val_loss: 2.0340 - val_accurac
y: 0.2728
Epoch 19/200
391/391 - 10s - loss: 2.0454 - accuracy: 0.2588 - val_loss: 2.0269 - val_accurac
y: 0.2985
Epoch 20/200
391/391 - 10s - loss: 2.0436 - accuracy: 0.2609 - val_loss: 2.0040 - val_accurac
```

```
y: 0.2928
Epoch 21/200
391/391 - 10s - loss: 2.0425 - accuracy: 0.2616 - val_loss: 2.0261 - val_accurac
y: 0.2699
Epoch 22/200
391/391 - 10s - loss: 2.0405 - accuracy: 0.2607 - val_loss: 2.0562 - val_accurac
y: 0.2820
Epoch 23/200
391/391 - 10s - loss: 2.0484 - accuracy: 0.2564 - val_loss: 2.0211 - val_accurac
y: 0.3059
Epoch 24/200
391/391 - 10s - loss: 2.0420 - accuracy: 0.2581 - val_loss: 2.0115 - val_accurac
y: 0.3274
Epoch 25/200
391/391 - 10s - loss: 2.0394 - accuracy: 0.2631 - val_loss: 2.0130 - val_accurac
y: 0.3121
Epoch 26/200
391/391 - 10s - loss: 2.0386 - accuracy: 0.2638 - val_loss: 2.0068 - val_accurac
y: 0.3004
Epoch 27/200
391/391 - 10s - loss: 2.0407 - accuracy: 0.2616 - val_loss: 2.0523 - val_accurac
y: 0.2920
Epoch 28/200
391/391 - 10s - loss: 2.0387 - accuracy: 0.2592 - val_loss: 2.0196 - val_accurac
y: 0.2976
Epoch 29/200
391/391 - 10s - loss: 2.0403 - accuracy: 0.2631 - val_loss: 2.0325 - val_accurac
y: 0.3039
Epoch 30/200
391/391 - 10s - loss: 2.0335 - accuracy: 0.2630 - val_loss: 2.0200 - val_accurac
y: 0.3146
Epoch 31/200
391/391 - 9s - loss: 2.0339 - accuracy: 0.2655 - val_loss: 2.0341 - val_accurac
y: 0.3035
Epoch 32/200
391/391 - 10s - loss: 2.0330 - accuracy: 0.2637 - val_loss: 2.0640 - val_accurac
y: 0.2584
Epoch 33/200
391/391 - 10s - loss: 2.0331 - accuracy: 0.2661 - val_loss: 2.0700 - val_accurac
y: 0.2528
Epoch 34/200
391/391 - 9s - loss: 2.0309 - accuracy: 0.2629 - val_loss: 2.0260 - val_accurac
y: 0.2919
Epoch 35/200
391/391 - 8s - loss: 2.0339 - accuracy: 0.2655 - val_loss: 2.0055 - val_accurac
y: 0.3211
Epoch 36/200
391/391 - 10s - loss: 2.0345 - accuracy: 0.2642 - val_loss: 2.0315 - val_accurac
y: 0.3053
Epoch 37/200
391/391 - 10s - loss: 2.0280 - accuracy: 0.2648 - val_loss: 2.0441 - val_accurac
y: 0.2951
Epoch 38/200
391/391 - 10s - loss: 2.0301 - accuracy: 0.2678 - val_loss: 2.0030 - val_accurac
y: 0.2974
Epoch 39/200
391/391 - 10s - loss: 2.0327 - accuracy: 0.2645 - val_loss: 2.0246 - val_accurac
y: 0.2979
Epoch 40/200
```

```
391/391 - 10s - loss: 2.0300 - accuracy: 0.2663 - val_loss: 2.0716 - val_accurac
y: 0.2650
Epoch 41/200
391/391 - 10s - loss: 2.0304 - accuracy: 0.2649 - val_loss: 2.0480 - val_accurac
y: 0.2732
Epoch 42/200
391/391 - 10s - loss: 2.0322 - accuracy: 0.2675 - val_loss: 2.0305 - val_accurac
y: 0.3143
Epoch 43/200
391/391 - 10s - loss: 2.0369 - accuracy: 0.2622 - val_loss: 2.0154 - val_accurac
y: 0.3205
Epoch 44/200
391/391 - 10s - loss: 2.0316 - accuracy: 0.2643 - val_loss: 2.0116 - val_accurac
y: 0.3042
Epoch 45/200
391/391 - 10s - loss: 2.0352 - accuracy: 0.2629 - val_loss: 2.0031 - val_accurac
y: 0.3150
Epoch 46/200
391/391 - 10s - loss: 2.0342 - accuracy: 0.2613 - val_loss: 2.0710 - val_accurac
y: 0.2459
Epoch 47/200
391/391 - 10s - loss: 2.0439 - accuracy: 0.2556 - val_loss: 2.0313 - val_accurac
y: 0.3187
Epoch 48/200
391/391 - 10s - loss: 2.0359 - accuracy: 0.2608 - val_loss: 2.0481 - val_accurac
y: 0.3090
Epoch 49/200
391/391 - 10s - loss: 2.0388 - accuracy: 0.2583 - val_loss: 2.0174 - val_accurac
y: 0.2835
Epoch 50/200
391/391 - 10s - loss: 2.0378 - accuracy: 0.2607 - val_loss: 2.0639 - val_accurac
y: 0.2801
Epoch 51/200
391/391 - 10s - loss: 2.0394 - accuracy: 0.2581 - val_loss: 2.0410 - val_accurac
y: 0.2723
Epoch 52/200
391/391 - 10s - loss: 2.0373 - accuracy: 0.2607 - val_loss: 2.0431 - val_accurac
y: 0.2528
Epoch 53/200
391/391 - 10s - loss: 2.0345 - accuracy: 0.2607 - val_loss: 2.0369 - val_accurac
y: 0.2835
Epoch 54/200
391/391 - 10s - loss: 2.0381 - accuracy: 0.2588 - val_loss: 2.0380 - val_accurac
y: 0.2833
Epoch 55/200
391/391 - 10s - loss: 2.0393 - accuracy: 0.2582 - val_loss: 2.0250 - val_accurac
y: 0.2917
Epoch 56/200
391/391 - 10s - loss: 2.0340 - accuracy: 0.2618 - val_loss: 2.0584 - val_accurac
y: 0.2748
Epoch 57/200
391/391 - 10s - loss: 2.0398 - accuracy: 0.2589 - val_loss: 2.0496 - val_accurac
y: 0.2855
Epoch 58/200
391/391 - 10s - loss: 2.0375 - accuracy: 0.2606 - val_loss: 2.0210 - val_accurac
y: 0.2895
Epoch 59/200
391/391 - 10s - loss: 2.0406 - accuracy: 0.2571 - val_loss: 2.0262 - val_accurac
y: 0.2916
```

```
Epoch 60/200
391/391 - 10s - loss: 2.0370 - accuracy: 0.2593 - val_loss: 2.0234 - val_accurac
y: 0.3101
Epoch 61/200
391/391 - 10s - loss: 2.0376 - accuracy: 0.2599 - val_loss: 2.0330 - val_accurac
y: 0.2997
Epoch 62/200
391/391 - 10s - loss: 2.0348 - accuracy: 0.2613 - val_loss: 2.0389 - val_accurac
y: 0.2870
Epoch 63/200
391/391 - 10s - loss: 2.0394 - accuracy: 0.2586 - val_loss: 2.0175 - val_accurac
y: 0.2948
Epoch 64/200
391/391 - 10s - loss: 2.0432 - accuracy: 0.2579 - val_loss: 2.0943 - val_accurac
y: 0.2541
Epoch 65/200
391/391 - 10s - loss: 2.0388 - accuracy: 0.2597 - val_loss: 2.0153 - val_accurac
y: 0.3011
Epoch 66/200
391/391 - 10s - loss: 2.0392 - accuracy: 0.2591 - val_loss: 2.0700 - val_accurac
y: 0.2722
Epoch 67/200
391/391 - 10s - loss: 2.0373 - accuracy: 0.2625 - val_loss: 2.0269 - val_accurac
y: 0.2738
Epoch 68/200
391/391 - 10s - loss: 2.0370 - accuracy: 0.2596 - val_loss: 2.0521 - val_accurac
y: 0.2780
Epoch 69/200
391/391 - 10s - loss: 2.0368 - accuracy: 0.2629 - val_loss: 2.0041 - val_accurac
y: 0.3086
Epoch 70/200
391/391 - 10s - loss: 2.0406 - accuracy: 0.2572 - val_loss: 2.0329 - val_accurac
y: 0.3006
Epoch 71/200
391/391 - 10s - loss: 2.0330 - accuracy: 0.2603 - val_loss: 2.0392 - val_accurac
y: 0.3070
Epoch 72/200
391/391 - 9s - loss: 2.0343 - accuracy: 0.2644 - val_loss: 2.0114 - val_accurac
y: 0.3019
Epoch 73/200
391/391 - 10s - loss: 2.0414 - accuracy: 0.2597 - val_loss: 2.0360 - val_accurac
y: 0.3024
Epoch 74/200
391/391 - 10s - loss: 2.0352 - accuracy: 0.2613 - val_loss: 2.0359 - val_accurac
y: 0.2816
Epoch 75/200
391/391 - 10s - loss: 2.0516 - accuracy: 0.2523 - val_loss: 1.9997 - val_accurac
y: 0.2949
Epoch 76/200
391/391 - 10s - loss: 2.0470 - accuracy: 0.2526 - val_loss: 2.0376 - val_accurac
y: 0.2733
Epoch 77/200
391/391 - 10s - loss: 2.0468 - accuracy: 0.2517 - val_loss: 2.0712 - val_accurac
y: 0.2695
Epoch 78/200
391/391 - 11s - loss: 2.0580 - accuracy: 0.2407 - val_loss: 2.0533 - val_accurac
y: 0.2507
Epoch 79/200
391/391 - 11s - loss: 2.0604 - accuracy: 0.2381 - val_loss: 2.0699 - val_accurac
```

```
y: 0.2538
Epoch 80/200
391/391 - 10s - loss: 2.0526 - accuracy: 0.2410 - val_loss: 2.0748 - val_accurac
y: 0.2536
Epoch 81/200
391/391 - 10s - loss: 2.0635 - accuracy: 0.2357 - val_loss: 2.0836 - val_accurac
y: 0.2384
Epoch 82/200
391/391 - 10s - loss: 2.0554 - accuracy: 0.2382 - val_loss: 2.0600 - val_accurac
y: 0.2685
Epoch 83/200
391/391 - 10s - loss: 2.0530 - accuracy: 0.2384 - val_loss: 2.0404 - val_accurac
y: 0.2846
Epoch 84/200
391/391 - 10s - loss: 2.0518 - accuracy: 0.2394 - val_loss: 2.0555 - val_accurac
y: 0.2739
Epoch 85/200
391/391 - 10s - loss: 2.0510 - accuracy: 0.2387 - val_loss: 2.0667 - val_accurac
y: 0.2717
Epoch 86/200
391/391 - 11s - loss: 2.0556 - accuracy: 0.2417 - val_loss: 2.0796 - val_accurac
y: 0.2519
Epoch 87/200
391/391 - 11s - loss: 2.0543 - accuracy: 0.2412 - val_loss: 2.0518 - val_accurac
y: 0.2791
Epoch 88/200
391/391 - 10s - loss: 2.0542 - accuracy: 0.2429 - val_loss: 2.0810 - val_accurac
y: 0.2581
Epoch 89/200
391/391 - 10s - loss: 2.0522 - accuracy: 0.2420 - val_loss: 2.0745 - val_accurac
y: 0.2613
Epoch 90/200
391/391 - 10s - loss: 2.0560 - accuracy: 0.2395 - val_loss: 2.0379 - val_accurac
y: 0.2859
Epoch 91/200
391/391 - 10s - loss: 2.0555 - accuracy: 0.2395 - val_loss: 2.0665 - val_accurac
y: 0.2466
Epoch 92/200
391/391 - 10s - loss: 2.0526 - accuracy: 0.2447 - val_loss: 2.0785 - val_accurac
y: 0.2537
Epoch 93/200
391/391 - 10s - loss: 2.0537 - accuracy: 0.2417 - val_loss: 2.0635 - val_accurac
y: 0.2630
Epoch 94/200
391/391 - 10s - loss: 2.0543 - accuracy: 0.2398 - val_loss: 2.0998 - val_accurac
y: 0.2667
Epoch 95/200
391/391 - 10s - loss: 2.0592 - accuracy: 0.2399 - val_loss: 2.0646 - val_accurac
y: 0.2862
Epoch 96/200
391/391 - 10s - loss: 2.0498 - accuracy: 0.2430 - val_loss: 2.0248 - val_accurac
y: 0.2963
Epoch 97/200
391/391 - 10s - loss: 2.0496 - accuracy: 0.2405 - val_loss: 2.0784 - val_accurac
y: 0.2747
Epoch 98/200
391/391 - 10s - loss: 2.0530 - accuracy: 0.2401 - val_loss: 2.0833 - val_accurac
y: 0.2549
Epoch 99/200
```

```
391/391 - 10s - loss: 2.0580 - accuracy: 0.2425 - val_loss: 2.0919 - val_accurac
y: 0.2500
Epoch 100/200
391/391 - 10s - loss: 2.0526 - accuracy: 0.2435 - val_loss: 2.0519 - val_accurac
y: 0.2846
Epoch 101/200
391/391 - 10s - loss: 2.0506 - accuracy: 0.2433 - val_loss: 2.0538 - val_accurac
y: 0.2651
Epoch 102/200
391/391 - 10s - loss: 2.0543 - accuracy: 0.2426 - val_loss: 2.0926 - val_accurac
y: 0.2530
Epoch 103/200
391/391 - 10s - loss: 2.0520 - accuracy: 0.2439 - val_loss: 2.1112 - val_accurac
y: 0.2530
Epoch 104/200
391/391 - 10s - loss: 2.0585 - accuracy: 0.2439 - val_loss: 2.0924 - val_accurac
y: 0.2443
Epoch 105/200
391/391 - 10s - loss: 2.0499 - accuracy: 0.2446 - val_loss: 2.0594 - val_accurac
y: 0.2746
Epoch 106/200
391/391 - 10s - loss: 2.0572 - accuracy: 0.2401 - val_loss: 2.0886 - val_accurac
y: 0.2531
Epoch 107/200
391/391 - 10s - loss: 2.0580 - accuracy: 0.2400 - val_loss: 2.0556 - val_accurac
y: 0.2879
Epoch 108/200
391/391 - 10s - loss: 2.0556 - accuracy: 0.2426 - val_loss: 2.0651 - val_accurac
y: 0.2633
Epoch 109/200
391/391 - 9s - loss: 2.0531 - accuracy: 0.2443 - val_loss: 2.0535 - val_accurac
y: 0.2774
Epoch 110/200
391/391 - 10s - loss: 2.0534 - accuracy: 0.2438 - val_loss: 2.0869 - val_accurac
y: 0.2685
Epoch 111/200
391/391 - 10s - loss: 2.0567 - accuracy: 0.2386 - val_loss: 2.0390 - val_accurac
y: 0.2718
Epoch 112/200
391/391 - 10s - loss: 2.0477 - accuracy: 0.2449 - val_loss: 2.0397 - val_accurac
y: 0.2837
Epoch 113/200
391/391 - 10s - loss: 2.0549 - accuracy: 0.2443 - val_loss: 2.0558 - val_accurac
y: 0.2637
Epoch 114/200
391/391 - 10s - loss: 2.0502 - accuracy: 0.2415 - val_loss: 2.0603 - val_accurac
y: 0.2638
Epoch 115/200
391/391 - 11s - loss: 2.0461 - accuracy: 0.2428 - val_loss: 2.0799 - val_accurac
y: 0.2394
Epoch 116/200
391/391 - 10s - loss: 2.0544 - accuracy: 0.2425 - val_loss: 2.0784 - val_accurac
y: 0.2413
Epoch 117/200
391/391 - 10s - loss: 2.0544 - accuracy: 0.2428 - val_loss: 2.0583 - val_accurac
y: 0.2795
Epoch 118/200
391/391 - 10s - loss: 2.0528 - accuracy: 0.2417 - val_loss: 2.0643 - val_accurac
y: 0.2631
```

```
Epoch 119/200
391/391 - 10s - loss: 2.0522 - accuracy: 0.2419 - val_loss: 2.0482 - val_accurac
y: 0.2854
Epoch 120/200
391/391 - 10s - loss: 2.0507 - accuracy: 0.2415 - val_loss: 2.0615 - val_accurac
y: 0.2756
Epoch 121/200
391/391 - 10s - loss: 2.0509 - accuracy: 0.2433 - val_loss: 2.0345 - val_accurac
y: 0.2620
Epoch 122/200
391/391 - 10s - loss: 2.0535 - accuracy: 0.2405 - val_loss: 2.0526 - val_accurac
y: 0.2770
Epoch 123/200
391/391 - 10s - loss: 2.0551 - accuracy: 0.2401 - val_loss: 2.0707 - val_accurac
y: 0.2851
Epoch 124/200
391/391 - 10s - loss: 2.0490 - accuracy: 0.2408 - val_loss: 2.0670 - val_accurac
y: 0.2681
Epoch 125/200
391/391 - 10s - loss: 2.0531 - accuracy: 0.2413 - val_loss: 2.0961 - val_accurac
y: 0.2482
Epoch 126/200
391/391 - 10s - loss: 2.0565 - accuracy: 0.2388 - val_loss: 2.0653 - val_accurac
y: 0.2552
Epoch 127/200
391/391 - 10s - loss: 2.0575 - accuracy: 0.2375 - val_loss: 2.0979 - val_accurac
y: 0.2549
Epoch 128/200
391/391 - 10s - loss: 2.0585 - accuracy: 0.2412 - val_loss: 2.0933 - val_accurac
y: 0.2505
Epoch 129/200
391/391 - 10s - loss: 2.0517 - accuracy: 0.2428 - val_loss: 2.0633 - val_accurac
y: 0.2767
Epoch 130/200
391/391 - 10s - loss: 2.0623 - accuracy: 0.2373 - val_loss: 2.1191 - val_accurac
y: 0.2609
Epoch 131/200
391/391 - 10s - loss: 2.0540 - accuracy: 0.2431 - val_loss: 2.0703 - val_accurac
y: 0.2594
Epoch 132/200
391/391 - 10s - loss: 2.0569 - accuracy: 0.2417 - val_loss: 2.0707 - val_accurac
y: 0.2654
Epoch 133/200
391/391 - 10s - loss: 2.0542 - accuracy: 0.2412 - val_loss: 2.0353 - val_accurac
y: 0.2765
Epoch 134/200
391/391 - 10s - loss: 2.0527 - accuracy: 0.2424 - val_loss: 2.0567 - val_accurac
y: 0.2552
Epoch 135/200
391/391 - 10s - loss: 2.0539 - accuracy: 0.2401 - val_loss: 2.0595 - val_accurac
y: 0.2475
Epoch 136/200
391/391 - 10s - loss: 2.0607 - accuracy: 0.2386 - val_loss: 2.0544 - val_accurac
y: 0.2807
Epoch 137/200
391/391 - 10s - loss: 2.0521 - accuracy: 0.2399 - val_loss: 2.0593 - val_accurac
y: 0.2543
Epoch 138/200
391/391 - 10s - loss: 2.0617 - accuracy: 0.2372 - val_loss: 2.0712 - val_accurac
```

```
y: 0.2664
Epoch 139/200
391/391 - 10s - loss: 2.0528 - accuracy: 0.2377 - val_loss: 2.0593 - val_accurac
y: 0.2840
Epoch 140/200
391/391 - 10s - loss: 2.0516 - accuracy: 0.2407 - val_loss: 2.0889 - val_accurac
y: 0.2558
Epoch 141/200
391/391 - 10s - loss: 2.0553 - accuracy: 0.2400 - val_loss: 2.0527 - val_accurac
y: 0.2660
Epoch 142/200
391/391 - 10s - loss: 2.0525 - accuracy: 0.2412 - val_loss: 2.0528 - val_accurac
y: 0.2626
Epoch 143/200
391/391 - 10s - loss: 2.0578 - accuracy: 0.2400 - val_loss: 2.0292 - val_accurac
y: 0.2732
Epoch 144/200
391/391 - 10s - loss: 2.0521 - accuracy: 0.2437 - val_loss: 2.0854 - val_accurac
y: 0.2503
Epoch 145/200
391/391 - 9s - loss: 2.0538 - accuracy: 0.2401 - val_loss: 2.0484 - val_accurac
y: 0.2822
Epoch 146/200
391/391 - 8s - loss: 2.0540 - accuracy: 0.2385 - val_loss: 2.0418 - val_accurac
y: 0.2650
Epoch 147/200
391/391 - 10s - loss: 2.0526 - accuracy: 0.2388 - val_loss: 2.0999 - val_accurac
y: 0.2558
Epoch 148/200
391/391 - 10s - loss: 2.0522 - accuracy: 0.2428 - val_loss: 2.0691 - val_accurac
y: 0.2754
Epoch 149/200
391/391 - 10s - loss: 2.0558 - accuracy: 0.2386 - val_loss: 2.0759 - val_accurac
y: 0.2718
Epoch 150/200
391/391 - 10s - loss: 2.0565 - accuracy: 0.2405 - val_loss: 2.0547 - val_accurac
y: 0.2586
Epoch 151/200
391/391 - 10s - loss: 2.0505 - accuracy: 0.2421 - val_loss: 2.1106 - val_accurac
y: 0.2495
Epoch 152/200
391/391 - 10s - loss: 2.0590 - accuracy: 0.2368 - val_loss: 2.0352 - val_accurac
y: 0.2804
Epoch 153/200
391/391 - 10s - loss: 2.0533 - accuracy: 0.2385 - val_loss: 2.1206 - val_accurac
y: 0.2409
Epoch 154/200
391/391 - 10s - loss: 2.0536 - accuracy: 0.2402 - val_loss: 2.0588 - val_accurac
y: 0.2605
Epoch 155/200
391/391 - 10s - loss: 2.0580 - accuracy: 0.2372 - val_loss: 2.0694 - val_accurac
y: 0.2615
Epoch 156/200
391/391 - 10s - loss: 2.0514 - accuracy: 0.2404 - val_loss: 2.0488 - val_accurac
y: 0.2581
Epoch 157/200
391/391 - 10s - loss: 2.0519 - accuracy: 0.2406 - val_loss: 2.1128 - val_accurac
y: 0.2396
Epoch 158/200
```

```
391/391 - 10s - loss: 2.0507 - accuracy: 0.2409 - val_loss: 2.0577 - val_accurac
y: 0.2691
Epoch 159/200
391/391 - 10s - loss: 2.0588 - accuracy: 0.2375 - val_loss: 2.0488 - val_accurac
y: 0.2662
Epoch 160/200
391/391 - 10s - loss: 2.0592 - accuracy: 0.2378 - val_loss: 2.0737 - val_accurac
y: 0.2668
Epoch 161/200
391/391 - 10s - loss: 2.0501 - accuracy: 0.2432 - val_loss: 2.0654 - val_accurac
y: 0.2740
Epoch 162/200
391/391 - 10s - loss: 2.0580 - accuracy: 0.2366 - val_loss: 2.1072 - val_accurac
y: 0.2417
Epoch 163/200
391/391 - 10s - loss: 2.0612 - accuracy: 0.2353 - val_loss: 2.0561 - val_accurac
y: 0.2437
Epoch 164/200
391/391 - 10s - loss: 2.0532 - accuracy: 0.2384 - val_loss: 2.0580 - val_accurac
y: 0.2771
Epoch 165/200
391/391 - 10s - loss: 2.0575 - accuracy: 0.2384 - val_loss: 2.1004 - val_accurac
y: 0.2349
Epoch 166/200
391/391 - 10s - loss: 2.0574 - accuracy: 0.2383 - val_loss: 2.0721 - val_accurac
y: 0.2639
Epoch 167/200
391/391 - 10s - loss: 2.0514 - accuracy: 0.2420 - val_loss: 2.0914 - val_accurac
y: 0.2443
Epoch 168/200
391/391 - 10s - loss: 2.0521 - accuracy: 0.2426 - val_loss: 2.0749 - val_accurac
y: 0.2589
Epoch 169/200
391/391 - 10s - loss: 2.0505 - accuracy: 0.2395 - val_loss: 2.0774 - val_accurac
y: 0.2531
Epoch 170/200
391/391 - 10s - loss: 2.0531 - accuracy: 0.2383 - val_loss: 2.0432 - val_accurac
y: 0.2816
Epoch 171/200
391/391 - 10s - loss: 2.0692 - accuracy: 0.2313 - val_loss: 2.0701 - val_accurac
y: 0.2586
Epoch 172/200
391/391 - 10s - loss: 2.0553 - accuracy: 0.2364 - val_loss: 2.0430 - val_accurac
y: 0.2836
Epoch 173/200
391/391 - 10s - loss: 2.0525 - accuracy: 0.2374 - val_loss: 2.0614 - val_accurac
y: 0.2422
Epoch 174/200
391/391 - 10s - loss: 2.0585 - accuracy: 0.2359 - val_loss: 2.0482 - val_accurac
y: 0.2772
Epoch 175/200
391/391 - 10s - loss: 2.0559 - accuracy: 0.2380 - val_loss: 2.0714 - val_accurac
y: 0.2595
Epoch 176/200
391/391 - 10s - loss: 2.0509 - accuracy: 0.2408 - val_loss: 2.0773 - val_accurac
y: 0.2522
Epoch 177/200
391/391 - 10s - loss: 2.0552 - accuracy: 0.2377 - val_loss: 2.0982 - val_accurac
y: 0.2591
```

```
Epoch 178/200
391/391 - 10s - loss: 2.0601 - accuracy: 0.2388 - val_loss: 2.0942 - val_accurac
y: 0.2433
Epoch 179/200
391/391 - 10s - loss: 2.0506 - accuracy: 0.2441 - val_loss: 2.0766 - val_accurac
y: 0.2721
Epoch 180/200
391/391 - 10s - loss: 2.0546 - accuracy: 0.2405 - val_loss: 2.0594 - val_accurac
y: 0.2806
Epoch 181/200
391/391 - 10s - loss: 2.0527 - accuracy: 0.2400 - val_loss: 2.0907 - val_accurac
y: 0.2382
Epoch 182/200
391/391 - 9s - loss: 2.0746 - accuracy: 0.2316 - val_loss: 2.0777 - val_accurac
y: 0.2686
Epoch 183/200
391/391 - 9s - loss: 2.0752 - accuracy: 0.2279 - val_loss: 2.0699 - val_accurac
y: 0.2623
Epoch 184/200
391/391 - 10s - loss: 2.0655 - accuracy: 0.2318 - val_loss: 2.0542 - val_accurac
y: 0.2730
Epoch 185/200
391/391 - 10s - loss: 2.0575 - accuracy: 0.2406 - val_loss: 2.0699 - val_accurac
y: 0.2734
Epoch 186/200
391/391 - 10s - loss: 2.0611 - accuracy: 0.2398 - val_loss: 2.0587 - val_accurac
y: 0.2597
Epoch 187/200
391/391 - 10s - loss: 2.0691 - accuracy: 0.2310 - val_loss: 2.0597 - val_accurac
y: 0.2741
Epoch 188/200
391/391 - 10s - loss: 2.0751 - accuracy: 0.2289 - val_loss: 2.0812 - val_accurac
y: 0.2456
Epoch 189/200
391/391 - 10s - loss: 2.0581 - accuracy: 0.2378 - val_loss: 2.0849 - val_accurac
y: 0.2620
Epoch 190/200
391/391 - 10s - loss: 2.0581 - accuracy: 0.2386 - val_loss: 2.0629 - val_accurac
y: 0.2352
Epoch 191/200
391/391 - 10s - loss: 2.0545 - accuracy: 0.2391 - val_loss: 2.0646 - val_accurac
y: 0.2714
Epoch 192/200
391/391 - 10s - loss: 2.0566 - accuracy: 0.2388 - val_loss: 2.0929 - val_accurac
y: 0.2489
Epoch 193/200
391/391 - 10s - loss: 2.0534 - accuracy: 0.2402 - val_loss: 2.0788 - val_accurac
y: 0.2581
Epoch 194/200
391/391 - 10s - loss: 2.0545 - accuracy: 0.2394 - val_loss: 2.0787 - val_accurac
y: 0.2589
Epoch 195/200
391/391 - 10s - loss: 2.0523 - accuracy: 0.2378 - val_loss: 2.0798 - val_accurac
y: 0.2575
Epoch 196/200
391/391 - 10s - loss: 2.0483 - accuracy: 0.2416 - val_loss: 2.0759 - val_accurac
y: 0.2621
Epoch 197/200
391/391 - 10s - loss: 2.0522 - accuracy: 0.2399 - val_loss: 2.0687 - val_accurac
```

```
y: 0.2625
Epoch 198/200
391/391 - 10s - loss: 2.0539 - accuracy: 0.2419 - val_loss: 2.0500 - val_accurac
y: 0.2675
Epoch 199/200
391/391 - 10s - loss: 2.0543 - accuracy: 0.2386 - val_loss: 2.0753 - val_accurac
y: 0.2771
Epoch 200/200
391/391 - 10s - loss: 2.0536 - accuracy: 0.2392 - val_loss: 2.0737 - val_accurac
y: 0.2396
```

Adadelta

In [8]:
```
model_adadelta_dropout=create_model()
model_adadelta_dropout.compile(loss='categorical_crossentropy', optimizer='Adadelta', metrics=['accuracy'])
history_adadelta_dropout = model_adadelta_dropout.fit(X_train, y_train, batch_size=128, epochs=200, validation_data=(X_test,y_test),shuffle=True,verbose=2)
```

```
Epoch 1/200
391/391 - 8s - loss: 42.9865 - accuracy: 0.1048 - val_loss: 42.0843 - val_accura
cy: 0.1269
Epoch 2/200
391/391 - 7s - loss: 42.3874 - accuracy: 0.1131 - val_loss: 41.7481 - val_accura
cy: 0.1506
Epoch 3/200
391/391 - 7s - loss: 41.9228 - accuracy: 0.1232 - val_loss: 41.4063 - val_accura
cy: 0.1746
Epoch 4/200
391/391 - 7s - loss: 41.5081 - accuracy: 0.1335 - val_loss: 41.0618 - val_accura
cy: 0.1979
Epoch 5/200
391/391 - 7s - loss: 41.1138 - accuracy: 0.1407 - val_loss: 40.7187 - val_accura
cy: 0.2091
Epoch 6/200
391/391 - 7s - loss: 40.7366 - accuracy: 0.1486 - val_loss: 40.3733 - val_accura
cy: 0.2207
Epoch 7/200
391/391 - 7s - loss: 40.3627 - accuracy: 0.1563 - val_loss: 40.0263 - val_accura
cy: 0.2266
Epoch 8/200
391/391 - 7s - loss: 39.9953 - accuracy: 0.1626 - val_loss: 39.6742 - val_accura
cy: 0.2395
Epoch 9/200
391/391 - 7s - loss: 39.6249 - accuracy: 0.1697 - val_loss: 39.3199 - val_accura
cy: 0.2437
Epoch 10/200
391/391 - 7s - loss: 39.2651 - accuracy: 0.1723 - val_loss: 38.9633 - val_accura
cy: 0.2512
Epoch 11/200
391/391 - 7s - loss: 38.9002 - accuracy: 0.1770 - val_loss: 38.6051 - val_accura
cy: 0.2554
Epoch 12/200
391/391 - 7s - loss: 38.5339 - accuracy: 0.1829 - val_loss: 38.2467 - val_accura
cy: 0.2600
Epoch 13/200
391/391 - 7s - loss: 38.1720 - accuracy: 0.1861 - val_loss: 37.8874 - val_accura
cy: 0.2652
Epoch 14/200
391/391 - 7s - loss: 37.8109 - accuracy: 0.1897 - val_loss: 37.5295 - val_accura
cy: 0.2679
Epoch 15/200
391/391 - 7s - loss: 37.4552 - accuracy: 0.1902 - val_loss: 37.1719 - val_accura
cy: 0.2712
Epoch 16/200
391/391 - 7s - loss: 37.0927 - accuracy: 0.1967 - val_loss: 36.8154 - val_accura
cy: 0.2757
Epoch 17/200
391/391 - 7s - loss: 36.7383 - accuracy: 0.1981 - val_loss: 36.4603 - val_accura
cy: 0.2817
Epoch 18/200
391/391 - 7s - loss: 36.3814 - accuracy: 0.2022 - val_loss: 36.1077 - val_accura
cy: 0.2845
Epoch 19/200
391/391 - 7s - loss: 36.0300 - accuracy: 0.2045 - val_loss: 35.7571 - val_accura
cy: 0.2885
Epoch 20/200
391/391 - 7s - loss: 35.6784 - accuracy: 0.2072 - val_loss: 35.4083 - val_accura
```

```
cy: 0.2901
Epoch 21/200
391/391 - 7s - loss: 35.3287 - accuracy: 0.2120 - val_loss: 35.0615 - val_accura
cy: 0.2918
Epoch 22/200
391/391 - 7s - loss: 34.9825 - accuracy: 0.2140 - val_loss: 34.7185 - val_accura
cy: 0.2892
Epoch 23/200
391/391 - 7s - loss: 34.6437 - accuracy: 0.2185 - val_loss: 34.3779 - val_accura
cy: 0.2939
Epoch 24/200
391/391 - 7s - loss: 34.3036 - accuracy: 0.2186 - val_loss: 34.0402 - val_accura
cy: 0.2951
Epoch 25/200
391/391 - 7s - loss: 33.9685 - accuracy: 0.2204 - val_loss: 33.7063 - val_accura
cy: 0.2973
Epoch 26/200
391/391 - 6s - loss: 33.6348 - accuracy: 0.2215 - val_loss: 33.3742 - val_accura
cy: 0.2999
Epoch 27/200
391/391 - 6s - loss: 33.3067 - accuracy: 0.2247 - val_loss: 33.0454 - val_accura
cy: 0.3034
Epoch 28/200
391/391 - 7s - loss: 32.9806 - accuracy: 0.2270 - val_loss: 32.7208 - val_accura
cy: 0.3037
Epoch 29/200
391/391 - 7s - loss: 32.6557 - accuracy: 0.2284 - val_loss: 32.3990 - val_accura
cy: 0.3055
Epoch 30/200
391/391 - 7s - loss: 32.3390 - accuracy: 0.2321 - val_loss: 32.0816 - val_accura
cy: 0.3062
Epoch 31/200
391/391 - 7s - loss: 32.0242 - accuracy: 0.2317 - val_loss: 31.7665 - val_accura
cy: 0.3077
Epoch 32/200
391/391 - 7s - loss: 31.7090 - accuracy: 0.2345 - val_loss: 31.4535 - val_accura
cy: 0.3091
Epoch 33/200
391/391 - 7s - loss: 31.3957 - accuracy: 0.2358 - val_loss: 31.1447 - val_accura
cy: 0.3096
Epoch 34/200
391/391 - 7s - loss: 31.0871 - accuracy: 0.2395 - val_loss: 30.8377 - val_accura
cy: 0.3112
Epoch 35/200
391/391 - 7s - loss: 30.7813 - accuracy: 0.2419 - val_loss: 30.5352 - val_accura
cy: 0.3115
Epoch 36/200
391/391 - 7s - loss: 30.4875 - accuracy: 0.2420 - val_loss: 30.2359 - val_accura
cy: 0.3133
Epoch 37/200
391/391 - 7s - loss: 30.1852 - accuracy: 0.2460 - val_loss: 29.9391 - val_accura
cy: 0.3145
Epoch 38/200
391/391 - 7s - loss: 29.8886 - accuracy: 0.2499 - val_loss: 29.6466 - val_accura
cy: 0.3144
Epoch 39/200
391/391 - 7s - loss: 29.6008 - accuracy: 0.2433 - val_loss: 29.3572 - val_accura
cy: 0.3157
Epoch 40/200
```

```
391/391 - 7s - loss: 29.3127 - accuracy: 0.2509 - val_loss: 29.0704 - val_accura
cy: 0.3174
Epoch 41/200
391/391 - 7s - loss: 29.0285 - accuracy: 0.2481 - val_loss: 28.7868 - val_accura
cy: 0.3200
Epoch 42/200
391/391 - 7s - loss: 28.7448 - accuracy: 0.2524 - val_loss: 28.5061 - val_accura
cy: 0.3210
Epoch 43/200
391/391 - 7s - loss: 28.4649 - accuracy: 0.2509 - val_loss: 28.2290 - val_accura
cy: 0.3205
Epoch 44/200
391/391 - 7s - loss: 28.1900 - accuracy: 0.2538 - val_loss: 27.9550 - val_accura
cy: 0.3225
Epoch 45/200
391/391 - 7s - loss: 27.9170 - accuracy: 0.2531 - val_loss: 27.6827 - val_accura
cy: 0.3239
Epoch 46/200
391/391 - 7s - loss: 27.6440 - accuracy: 0.2563 - val_loss: 27.4139 - val_accura
cy: 0.3238
Epoch 47/200
391/391 - 7s - loss: 27.3786 - accuracy: 0.2603 - val_loss: 27.1479 - val_accura
cy: 0.3242
Epoch 48/200
391/391 - 7s - loss: 27.1160 - accuracy: 0.2584 - val_loss: 26.8858 - val_accura
cy: 0.3259
Epoch 49/200
391/391 - 7s - loss: 26.8556 - accuracy: 0.2615 - val_loss: 26.6268 - val_accura
cy: 0.3266
Epoch 50/200
391/391 - 7s - loss: 26.5942 - accuracy: 0.2629 - val_loss: 26.3695 - val_accura
cy: 0.3271
Epoch 51/200
391/391 - 7s - loss: 26.3397 - accuracy: 0.2640 - val_loss: 26.1152 - val_accura
cy: 0.3296
Epoch 52/200
391/391 - 7s - loss: 26.0889 - accuracy: 0.2609 - val_loss: 25.8653 - val_accura
cy: 0.3297
Epoch 53/200
391/391 - 7s - loss: 25.8402 - accuracy: 0.2662 - val_loss: 25.6172 - val_accura
cy: 0.3288
Epoch 54/200
391/391 - 7s - loss: 25.5922 - accuracy: 0.2666 - val_loss: 25.3716 - val_accura
cy: 0.3308
Epoch 55/200
391/391 - 7s - loss: 25.3458 - accuracy: 0.2668 - val_loss: 25.1282 - val_accura
cy: 0.3316
Epoch 56/200
391/391 - 7s - loss: 25.1055 - accuracy: 0.2675 - val_loss: 24.8876 - val_accura
cy: 0.3331
Epoch 57/200
391/391 - 7s - loss: 24.8660 - accuracy: 0.2724 - val_loss: 24.6505 - val_accura
cy: 0.3343
Epoch 58/200
391/391 - 7s - loss: 24.6312 - accuracy: 0.2714 - val_loss: 24.4161 - val_accura
cy: 0.3351
Epoch 59/200
391/391 - 7s - loss: 24.3959 - accuracy: 0.2734 - val_loss: 24.1828 - val_accura
cy: 0.3364
```

```
Epoch 60/200
391/391 - 7s - loss: 24.1698 - accuracy: 0.2737 - val_loss: 23.9534 - val_accura
cy: 0.3369
Epoch 61/200
391/391 - 7s - loss: 23.9370 - accuracy: 0.2737 - val_loss: 23.7273 - val_accura
cy: 0.3365
Epoch 62/200
391/391 - 7s - loss: 23.7101 - accuracy: 0.2793 - val_loss: 23.5030 - val_accura
cy: 0.3383
Epoch 63/200
391/391 - 7s - loss: 23.4875 - accuracy: 0.2763 - val_loss: 23.2818 - val_accura
cy: 0.3382
Epoch 64/200
391/391 - 7s - loss: 23.2675 - accuracy: 0.2803 - val_loss: 23.0630 - val_accura
cy: 0.3382
Epoch 65/200
391/391 - 7s - loss: 23.0551 - accuracy: 0.2798 - val_loss: 22.8467 - val_accura
cy: 0.3409
Epoch 66/200
391/391 - 7s - loss: 22.8358 - accuracy: 0.2805 - val_loss: 22.6328 - val_accura
cy: 0.3403
Epoch 67/200
391/391 - 7s - loss: 22.6242 - accuracy: 0.2769 - val_loss: 22.4215 - val_accura
cy: 0.3412
Epoch 68/200
391/391 - 7s - loss: 22.4122 - accuracy: 0.2805 - val_loss: 22.2119 - val_accura
cy: 0.3426
Epoch 69/200
391/391 - 7s - loss: 22.2053 - accuracy: 0.2818 - val_loss: 22.0055 - val_accura
cy: 0.3420
Epoch 70/200
391/391 - 7s - loss: 22.0005 - accuracy: 0.2813 - val_loss: 21.8000 - val_accura
cy: 0.3432
Epoch 71/200
391/391 - 7s - loss: 21.7964 - accuracy: 0.2858 - val_loss: 21.5990 - val_accura
cy: 0.3430
Epoch 72/200
391/391 - 7s - loss: 21.5956 - accuracy: 0.2827 - val_loss: 21.3980 - val_accura
cy: 0.3448
Epoch 73/200
391/391 - 7s - loss: 21.3979 - accuracy: 0.2852 - val_loss: 21.2005 - val_accura
cy: 0.3463
Epoch 74/200
391/391 - 7s - loss: 21.1996 - accuracy: 0.2858 - val_loss: 21.0052 - val_accura
cy: 0.3440
Epoch 75/200
391/391 - 7s - loss: 21.0063 - accuracy: 0.2867 - val_loss: 20.8107 - val_accura
cy: 0.3471
Epoch 76/200
391/391 - 7s - loss: 20.8149 - accuracy: 0.2871 - val_loss: 20.6192 - val_accura
cy: 0.3481
Epoch 77/200
391/391 - 6s - loss: 20.6208 - accuracy: 0.2925 - val_loss: 20.4305 - val_accura
cy: 0.3487
Epoch 78/200
391/391 - 6s - loss: 20.4355 - accuracy: 0.2903 - val_loss: 20.2429 - val_accura
cy: 0.3488
Epoch 79/200
391/391 - 7s - loss: 20.2467 - accuracy: 0.2912 - val_loss: 20.0583 - val_accura
```

```
cy: 0.3493
Epoch 80/200
391/391 - 7s - loss: 20.0624 - accuracy: 0.2921 - val_loss: 19.8754 - val_accura
cy: 0.3511
Epoch 81/200
391/391 - 7s - loss: 19.8796 - accuracy: 0.2938 - val_loss: 19.6948 - val_accura
cy: 0.3509
Epoch 82/200
391/391 - 7s - loss: 19.7039 - accuracy: 0.2898 - val_loss: 19.5151 - val_accura
cy: 0.3531
Epoch 83/200
391/391 - 7s - loss: 19.5279 - accuracy: 0.2912 - val_loss: 19.3390 - val_accura
cy: 0.3517
Epoch 84/200
391/391 - 7s - loss: 19.3482 - accuracy: 0.2964 - val_loss: 19.1640 - val_accura
cy: 0.3516
Epoch 85/200
391/391 - 7s - loss: 19.1739 - accuracy: 0.2944 - val_loss: 18.9907 - val_accura
cy: 0.3542
Epoch 86/200
391/391 - 7s - loss: 18.9983 - accuracy: 0.2980 - val_loss: 18.8189 - val_accura
cy: 0.3543
Epoch 87/200
391/391 - 7s - loss: 18.8303 - accuracy: 0.2971 - val_loss: 18.6497 - val_accura
cy: 0.3550
Epoch 88/200
391/391 - 7s - loss: 18.6656 - accuracy: 0.2960 - val_loss: 18.4836 - val_accura
cy: 0.3553
Epoch 89/200
391/391 - 7s - loss: 18.4965 - accuracy: 0.3005 - val_loss: 18.3168 - val_accura
cy: 0.3565
Epoch 90/200
391/391 - 7s - loss: 18.3323 - accuracy: 0.2969 - val_loss: 18.1533 - val_accura
cy: 0.3580
Epoch 91/200
391/391 - 7s - loss: 18.1649 - accuracy: 0.3031 - val_loss: 17.9903 - val_accura
cy: 0.3576
Epoch 92/200
391/391 - 7s - loss: 18.0097 - accuracy: 0.2993 - val_loss: 17.8315 - val_accura
cy: 0.3573
Epoch 93/200
391/391 - 7s - loss: 17.8430 - accuracy: 0.3008 - val_loss: 17.6716 - val_accura
cy: 0.3580
Epoch 94/200
391/391 - 7s - loss: 17.6888 - accuracy: 0.3043 - val_loss: 17.5154 - val_accura
cy: 0.3600
Epoch 95/200
391/391 - 7s - loss: 17.5364 - accuracy: 0.3040 - val_loss: 17.3611 - val_accura
cy: 0.3598
Epoch 96/200
391/391 - 7s - loss: 17.3752 - accuracy: 0.3075 - val_loss: 17.2081 - val_accura
cy: 0.3598
Epoch 97/200
391/391 - 7s - loss: 17.2283 - accuracy: 0.3048 - val_loss: 17.0559 - val_accura
cy: 0.3620
Epoch 98/200
391/391 - 7s - loss: 17.0739 - accuracy: 0.3100 - val_loss: 16.9058 - val_accura
cy: 0.3616
Epoch 99/200
```

```
391/391 - 7s - loss: 16.9273 - accuracy: 0.3064 - val_loss: 16.7569 - val_accura
cy: 0.3625
Epoch 100/200
391/391 - 7s - loss: 16.7805 - accuracy: 0.3088 - val_loss: 16.6121 - val_accura
cy: 0.3622
Epoch 101/200
391/391 - 7s - loss: 16.6351 - accuracy: 0.3068 - val_loss: 16.4654 - val_accura
cy: 0.3635
Epoch 102/200
391/391 - 7s - loss: 16.4893 - accuracy: 0.3085 - val_loss: 16.3214 - val_accura
cy: 0.3648
Epoch 103/200
391/391 - 7s - loss: 16.3451 - accuracy: 0.3085 - val_loss: 16.1791 - val_accura
cy: 0.3657
Epoch 104/200
391/391 - 7s - loss: 16.2096 - accuracy: 0.3075 - val_loss: 16.0394 - val_accura
cy: 0.3638
Epoch 105/200
391/391 - 7s - loss: 16.0638 - accuracy: 0.3124 - val_loss: 15.9006 - val_accura
cy: 0.3663
Epoch 106/200
391/391 - 7s - loss: 15.9290 - accuracy: 0.3120 - val_loss: 15.7632 - val_accura
cy: 0.3668
Epoch 107/200
391/391 - 7s - loss: 15.7884 - accuracy: 0.3145 - val_loss: 15.6290 - val_accura
cy: 0.3659
Epoch 108/200
391/391 - 7s - loss: 15.6548 - accuracy: 0.3136 - val_loss: 15.4932 - val_accura
cy: 0.3672
Epoch 109/200
391/391 - 7s - loss: 15.5234 - accuracy: 0.3130 - val_loss: 15.3620 - val_accura
cy: 0.3662
Epoch 110/200
391/391 - 7s - loss: 15.3922 - accuracy: 0.3115 - val_loss: 15.2290 - val_accura
cy: 0.3689
Epoch 111/200
391/391 - 7s - loss: 15.2625 - accuracy: 0.3148 - val_loss: 15.0995 - val_accura
cy: 0.3687
Epoch 112/200
391/391 - 7s - loss: 15.1308 - accuracy: 0.3165 - val_loss: 14.9730 - val_accura
cy: 0.3693
Epoch 113/200
391/391 - 7s - loss: 15.0058 - accuracy: 0.3145 - val_loss: 14.8443 - val_accura
cy: 0.3692
Epoch 114/200
391/391 - 7s - loss: 14.8787 - accuracy: 0.3156 - val_loss: 14.7192 - val_accura
cy: 0.3710
Epoch 115/200
391/391 - 7s - loss: 14.7514 - accuracy: 0.3182 - val_loss: 14.5939 - val_accura
cy: 0.3713
Epoch 116/200
391/391 - 7s - loss: 14.6281 - accuracy: 0.3171 - val_loss: 14.4710 - val_accura
cy: 0.3718
Epoch 117/200
391/391 - 7s - loss: 14.5087 - accuracy: 0.3165 - val_loss: 14.3493 - val_accura
cy: 0.3729
Epoch 118/200
391/391 - 7s - loss: 14.3838 - accuracy: 0.3175 - val_loss: 14.2309 - val_accura
cy: 0.3705
```

```
Epoch 119/200
391/391 - 7s - loss: 14.2674 - accuracy: 0.3189 - val_loss: 14.1104 - val_accura
cy: 0.3719
Epoch 120/200
391/391 - 7s - loss: 14.1460 - accuracy: 0.3208 - val_loss: 13.9926 - val_accura
cy: 0.3716
Epoch 121/200
391/391 - 7s - loss: 14.0290 - accuracy: 0.3212 - val_loss: 13.8747 - val_accura
cy: 0.3740
Epoch 122/200
391/391 - 7s - loss: 13.9155 - accuracy: 0.3210 - val_loss: 13.7590 - val_accura
cy: 0.3736
Epoch 123/200
391/391 - 7s - loss: 13.8010 - accuracy: 0.3212 - val_loss: 13.6442 - val_accura
cy: 0.3756
Epoch 124/200
391/391 - 7s - loss: 13.6826 - accuracy: 0.3234 - val_loss: 13.5324 - val_accura
cy: 0.3760
Epoch 125/200
391/391 - 7s - loss: 13.5720 - accuracy: 0.3230 - val_loss: 13.4207 - val_accura
cy: 0.3750
Epoch 126/200
391/391 - 7s - loss: 13.4583 - accuracy: 0.3253 - val_loss: 13.3098 - val_accura
cy: 0.3738
Epoch 127/200
391/391 - 6s - loss: 13.3502 - accuracy: 0.3252 - val_loss: 13.1992 - val_accura
cy: 0.3770
Epoch 128/200
391/391 - 6s - loss: 13.2406 - accuracy: 0.3233 - val_loss: 13.0913 - val_accura
cy: 0.3756
Epoch 129/200
391/391 - 7s - loss: 13.1332 - accuracy: 0.3260 - val_loss: 12.9846 - val_accura
cy: 0.3755
Epoch 130/200
391/391 - 7s - loss: 13.0295 - accuracy: 0.3244 - val_loss: 12.8784 - val_accura
cy: 0.3774
Epoch 131/200
391/391 - 7s - loss: 12.9215 - accuracy: 0.3265 - val_loss: 12.7731 - val_accura
cy: 0.3781
Epoch 132/200
391/391 - 7s - loss: 12.8180 - accuracy: 0.3260 - val_loss: 12.6695 - val_accura
cy: 0.3776
Epoch 133/200
391/391 - 7s - loss: 12.7121 - accuracy: 0.3298 - val_loss: 12.5683 - val_accura
cy: 0.3771
Epoch 134/200
391/391 - 7s - loss: 12.6092 - accuracy: 0.3287 - val_loss: 12.4654 - val_accura
cy: 0.3775
Epoch 135/200
391/391 - 7s - loss: 12.5107 - accuracy: 0.3282 - val_loss: 12.3649 - val_accura
cy: 0.3784
Epoch 136/200
391/391 - 7s - loss: 12.4092 - accuracy: 0.3321 - val_loss: 12.2659 - val_accura
cy: 0.3786
Epoch 137/200
391/391 - 7s - loss: 12.3124 - accuracy: 0.3298 - val_loss: 12.1660 - val_accura
cy: 0.3817
Epoch 138/200
391/391 - 7s - loss: 12.2131 - accuracy: 0.3313 - val_loss: 12.0681 - val_accura
```

```
cy: 0.3812
Epoch 139/200
391/391 - 7s - loss: 12.1184 - accuracy: 0.3275 - val_loss: 11.9739 - val_accura
cy: 0.3803
Epoch 140/200
391/391 - 7s - loss: 12.0233 - accuracy: 0.3295 - val_loss: 11.8779 - val_accura
cy: 0.3812
Epoch 141/200
391/391 - 7s - loss: 11.9275 - accuracy: 0.3306 - val_loss: 11.7842 - val_accura
cy: 0.3818
Epoch 142/200
391/391 - 7s - loss: 11.8339 - accuracy: 0.3328 - val_loss: 11.6905 - val_accura
cy: 0.3834
Epoch 143/200
391/391 - 7s - loss: 11.7409 - accuracy: 0.3320 - val_loss: 11.5958 - val_accura
cy: 0.3850
Epoch 144/200
391/391 - 7s - loss: 11.6430 - accuracy: 0.3330 - val_loss: 11.5061 - val_accura
cy: 0.3821
Epoch 145/200
391/391 - 7s - loss: 11.5557 - accuracy: 0.3327 - val_loss: 11.4133 - val_accura
cy: 0.3836
Epoch 146/200
391/391 - 7s - loss: 11.4628 - accuracy: 0.3364 - val_loss: 11.3234 - val_accura
cy: 0.3841
Epoch 147/200
391/391 - 7s - loss: 11.3741 - accuracy: 0.3356 - val_loss: 11.2345 - val_accura
cy: 0.3846
Epoch 148/200
391/391 - 7s - loss: 11.2853 - accuracy: 0.3339 - val_loss: 11.1476 - val_accura
cy: 0.3826
Epoch 149/200
391/391 - 7s - loss: 11.2002 - accuracy: 0.3333 - val_loss: 11.0592 - val_accura
cy: 0.3859
Epoch 150/200
391/391 - 7s - loss: 11.1139 - accuracy: 0.3361 - val_loss: 10.9738 - val_accura
cy: 0.3852
Epoch 151/200
391/391 - 7s - loss: 11.0272 - accuracy: 0.3389 - val_loss: 10.8873 - val_accura
cy: 0.3877
Epoch 152/200
391/391 - 7s - loss: 10.9414 - accuracy: 0.3370 - val_loss: 10.8037 - val_accura
cy: 0.3878
Epoch 153/200
391/391 - 7s - loss: 10.8596 - accuracy: 0.3346 - val_loss: 10.7210 - val_accura
cy: 0.3869
Epoch 154/200
391/391 - 7s - loss: 10.7738 - accuracy: 0.3383 - val_loss: 10.6378 - val_accura
cy: 0.3873
Epoch 155/200
391/391 - 7s - loss: 10.6898 - accuracy: 0.3399 - val_loss: 10.5568 - val_accura
cy: 0.3867
Epoch 156/200
391/391 - 7s - loss: 10.6120 - accuracy: 0.3380 - val_loss: 10.4736 - val_accura
cy: 0.3894
Epoch 157/200
391/391 - 7s - loss: 10.5296 - accuracy: 0.3412 - val_loss: 10.3947 - val_accura
cy: 0.3887
Epoch 158/200
```

```
391/391 - 7s - loss: 10.4505 - accuracy: 0.3403 - val_loss: 10.3144 - val_accura
cy: 0.3899
Epoch 159/200
391/391 - 7s - loss: 10.3703 - accuracy: 0.3419 - val_loss: 10.2384 - val_accura
cy: 0.3866
Epoch 160/200
391/391 - 7s - loss: 10.2940 - accuracy: 0.3400 - val_loss: 10.1586 - val_accura
cy: 0.3892
Epoch 161/200
391/391 - 7s - loss: 10.2155 - accuracy: 0.3413 - val_loss: 10.0808 - val_accura
cy: 0.3891
Epoch 162/200
391/391 - 7s - loss: 10.1380 - accuracy: 0.3408 - val_loss: 10.0066 - val_accura
cy: 0.3889
Epoch 163/200
391/391 - 7s - loss: 10.0614 - accuracy: 0.3425 - val_loss: 9.9309 - val_accurac
y: 0.3893
Epoch 164/200
391/391 - 7s - loss: 9.9870 - accuracy: 0.3416 - val_loss: 9.8543 - val_accurac
y: 0.3915
Epoch 165/200
391/391 - 7s - loss: 9.9097 - accuracy: 0.3440 - val_loss: 9.7806 - val_accurac
y: 0.3924
Epoch 166/200
391/391 - 7s - loss: 9.8420 - accuracy: 0.3406 - val_loss: 9.7100 - val_accurac
y: 0.3900
Epoch 167/200
391/391 - 7s - loss: 9.7668 - accuracy: 0.3418 - val_loss: 9.6357 - val_accurac
y: 0.3913
Epoch 168/200
391/391 - 7s - loss: 9.6939 - accuracy: 0.3442 - val_loss: 9.5623 - val_accurac
y: 0.3924
Epoch 169/200
391/391 - 7s - loss: 9.6259 - accuracy: 0.3437 - val_loss: 9.4920 - val_accurac
y: 0.3913
Epoch 170/200
391/391 - 7s - loss: 9.5529 - accuracy: 0.3457 - val_loss: 9.4210 - val_accurac
y: 0.3931
Epoch 171/200
391/391 - 7s - loss: 9.4807 - accuracy: 0.3424 - val_loss: 9.3519 - val_accurac
y: 0.3927
Epoch 172/200
391/391 - 7s - loss: 9.4116 - accuracy: 0.3472 - val_loss: 9.2821 - val_accurac
y: 0.3935
Epoch 173/200
391/391 - 7s - loss: 9.3414 - accuracy: 0.3484 - val_loss: 9.2131 - val_accurac
y: 0.3953
Epoch 174/200
391/391 - 7s - loss: 9.2759 - accuracy: 0.3456 - val_loss: 9.1462 - val_accurac
y: 0.3948
Epoch 175/200
391/391 - 7s - loss: 9.2048 - accuracy: 0.3504 - val_loss: 9.0773 - val_accurac
y: 0.3961
Epoch 176/200
391/391 - 7s - loss: 9.1390 - accuracy: 0.3487 - val_loss: 9.0112 - val_accurac
y: 0.3949
Epoch 177/200
391/391 - 7s - loss: 9.0728 - accuracy: 0.3477 - val_loss: 8.9470 - val_accurac
y: 0.3948
```

```
Epoch 178/200
391/391 - 6s - loss: 9.0088 - accuracy: 0.3484 - val_loss: 8.8822 - val_accurac
y: 0.3938
Epoch 179/200
391/391 - 7s - loss: 8.9436 - accuracy: 0.3488 - val_loss: 8.8177 - val_accurac
y: 0.3937
Epoch 180/200
391/391 - 7s - loss: 8.8788 - accuracy: 0.3485 - val_loss: 8.7538 - val_accurac
y: 0.3952
Epoch 181/200
391/391 - 7s - loss: 8.8134 - accuracy: 0.3512 - val_loss: 8.6904 - val_accurac
y: 0.3935
Epoch 182/200
391/391 - 7s - loss: 8.7498 - accuracy: 0.3518 - val_loss: 8.6260 - val_accurac
y: 0.3985
Epoch 183/200
391/391 - 7s - loss: 8.6903 - accuracy: 0.3517 - val_loss: 8.5664 - val_accurac
y: 0.3953
Epoch 184/200
391/391 - 7s - loss: 8.6310 - accuracy: 0.3519 - val_loss: 8.5044 - val_accurac
y: 0.3958
Epoch 185/200
391/391 - 7s - loss: 8.5685 - accuracy: 0.3494 - val_loss: 8.4440 - val_accurac
y: 0.3968
Epoch 186/200
391/391 - 7s - loss: 8.5039 - accuracy: 0.3530 - val_loss: 8.3824 - val_accurac
y: 0.3976
Epoch 187/200
391/391 - 7s - loss: 8.4443 - accuracy: 0.3531 - val_loss: 8.3240 - val_accurac
y: 0.3968
Epoch 188/200
391/391 - 7s - loss: 8.3880 - accuracy: 0.3546 - val_loss: 8.2638 - val_accurac
y: 0.3986
Epoch 189/200
391/391 - 7s - loss: 8.3261 - accuracy: 0.3543 - val_loss: 8.2062 - val_accurac
y: 0.3973
Epoch 190/200
391/391 - 7s - loss: 8.2691 - accuracy: 0.3517 - val_loss: 8.1500 - val_accurac
y: 0.3972
Epoch 191/200
391/391 - 7s - loss: 8.2132 - accuracy: 0.3520 - val_loss: 8.0921 - val_accurac
y: 0.3982
Epoch 192/200
391/391 - 7s - loss: 8.1567 - accuracy: 0.3529 - val_loss: 8.0351 - val_accurac
y: 0.3986
Epoch 193/200
391/391 - 7s - loss: 8.0981 - accuracy: 0.3539 - val_loss: 7.9786 - val_accurac
y: 0.3981
Epoch 194/200
391/391 - 8s - loss: 8.0430 - accuracy: 0.3516 - val_loss: 7.9224 - val_accurac
y: 0.4004
Epoch 195/200
391/391 - 7s - loss: 7.9883 - accuracy: 0.3523 - val_loss: 7.8681 - val_accurac
y: 0.4006
Epoch 196/200
391/391 - 7s - loss: 7.9322 - accuracy: 0.3585 - val_loss: 7.8138 - val_accurac
y: 0.3997
Epoch 197/200
391/391 - 7s - loss: 7.8774 - accuracy: 0.3549 - val_loss: 7.7589 - val_accurac
```

```
y: 0.4010
Epoch 198/200
391/391 - 7s - loss: 7.8236 - accuracy: 0.3576 - val_loss: 7.7057 - val_accurac
y: 0.3994
Epoch 199/200
391/391 - 7s - loss: 7.7706 - accuracy: 0.3544 - val_loss: 7.6518 - val_accurac
y: 0.3996
Epoch 200/200
391/391 - 7s - loss: 7.7165 - accuracy: 0.3577 - val_loss: 7.6006 - val_accurac
y: 0.3985
```

Adam

In [9]:
```
model_adam_dropout=create_model()
model_adam_dropout.compile(loss='categorical_crossentropy', optimizer='Adam', met
rics=['accuracy'])
history_adam_dropout = model_adam_dropout.fit(X_train, y_train, batch_size=128, e
pochs=200, validation_data=(X_test,y_test),shuffle=True,verbose=2)
```

```
Epoch 1/200
391/391 - 7s - loss: 7.4007 - accuracy: 0.2194 - val_loss: 2.2090 - val_accurac
y: 0.2864
Epoch 2/200
391/391 - 7s - loss: 2.1813 - accuracy: 0.2389 - val_loss: 2.0407 - val_accurac
y: 0.3011
Epoch 3/200
391/391 - 7s - loss: 2.1266 - accuracy: 0.2353 - val_loss: 2.0098 - val_accurac
y: 0.3079
Epoch 4/200
391/391 - 7s - loss: 2.1159 - accuracy: 0.2314 - val_loss: 2.0562 - val_accurac
y: 0.2710
Epoch 5/200
391/391 - 7s - loss: 2.1315 - accuracy: 0.2213 - val_loss: 2.0826 - val_accurac
y: 0.2521
Epoch 6/200
391/391 - 7s - loss: 2.1171 - accuracy: 0.2135 - val_loss: 2.0798 - val_accurac
y: 0.2632
Epoch 7/200
391/391 - 6s - loss: 2.1153 - accuracy: 0.2148 - val_loss: 2.0650 - val_accurac
y: 0.2336
Epoch 8/200
391/391 - 7s - loss: 2.1086 - accuracy: 0.2157 - val_loss: 2.0706 - val_accurac
y: 0.2510
Epoch 9/200
391/391 - 7s - loss: 2.1217 - accuracy: 0.2091 - val_loss: 2.0888 - val_accurac
y: 0.2113
Epoch 10/200
391/391 - 6s - loss: 2.1082 - accuracy: 0.2139 - val_loss: 2.0663 - val_accurac
y: 0.2295
Epoch 11/200
391/391 - 7s - loss: 2.1034 - accuracy: 0.2157 - val_loss: 2.0729 - val_accurac
y: 0.2474
Epoch 12/200
391/391 - 7s - loss: 2.1043 - accuracy: 0.2172 - val_loss: 2.1004 - val_accurac
y: 0.2194
Epoch 13/200
391/391 - 7s - loss: 2.0995 - accuracy: 0.2170 - val_loss: 2.0661 - val_accurac
y: 0.2498
Epoch 14/200
391/391 - 7s - loss: 2.0988 - accuracy: 0.2182 - val_loss: 2.0750 - val_accurac
y: 0.2264
Epoch 15/200
391/391 - 7s - loss: 2.1033 - accuracy: 0.2174 - val_loss: 2.0985 - val_accurac
y: 0.2207
Epoch 16/200
391/391 - 7s - loss: 2.0993 - accuracy: 0.2188 - val_loss: 2.1027 - val_accurac
y: 0.2656
Epoch 17/200
391/391 - 7s - loss: 2.0965 - accuracy: 0.2227 - val_loss: 2.1114 - val_accurac
y: 0.2396
Epoch 18/200
391/391 - 6s - loss: 2.0924 - accuracy: 0.2215 - val_loss: 2.0949 - val_accurac
y: 0.2229
Epoch 19/200
391/391 - 7s - loss: 2.1133 - accuracy: 0.2142 - val_loss: 2.1025 - val_accurac
y: 0.2372
Epoch 20/200
391/391 - 7s - loss: 2.0973 - accuracy: 0.2223 - val_loss: 2.1169 - val_accurac
```

```
y: 0.2168
Epoch 21/200
391/391 - 7s - loss: 2.0979 - accuracy: 0.2205 - val_loss: 2.1108 - val_accurac
y: 0.2226
Epoch 22/200
391/391 - 6s - loss: 2.0943 - accuracy: 0.2233 - val_loss: 2.0811 - val_accurac
y: 0.2447
Epoch 23/200
391/391 - 7s - loss: 2.0923 - accuracy: 0.2216 - val_loss: 2.0660 - val_accurac
y: 0.2478
Epoch 24/200
391/391 - 7s - loss: 2.0908 - accuracy: 0.2215 - val_loss: 2.0739 - val_accurac
y: 0.2137
Epoch 25/200
391/391 - 7s - loss: 2.0960 - accuracy: 0.2198 - val_loss: 2.0814 - val_accurac
y: 0.2515
Epoch 26/200
391/391 - 6s - loss: 2.0968 - accuracy: 0.2207 - val_loss: 2.1025 - val_accurac
y: 0.2170
Epoch 27/200
391/391 - 7s - loss: 2.0921 - accuracy: 0.2201 - val_loss: 2.1379 - val_accurac
y: 0.2054
Epoch 28/200
391/391 - 6s - loss: 2.0904 - accuracy: 0.2234 - val_loss: 2.0559 - val_accurac
y: 0.2588
Epoch 29/200
391/391 - 6s - loss: 2.0976 - accuracy: 0.2191 - val_loss: 2.1049 - val_accurac
y: 0.2163
Epoch 30/200
391/391 - 6s - loss: 2.0920 - accuracy: 0.2201 - val_loss: 2.1087 - val_accurac
y: 0.2162
Epoch 31/200
391/391 - 7s - loss: 2.0856 - accuracy: 0.2235 - val_loss: 2.1218 - val_accurac
y: 0.2229
Epoch 32/200
391/391 - 7s - loss: 2.1129 - accuracy: 0.2140 - val_loss: 2.0951 - val_accurac
y: 0.2429
Epoch 33/200
391/391 - 7s - loss: 2.0968 - accuracy: 0.2171 - val_loss: 2.0425 - val_accurac
y: 0.2580
Epoch 34/200
391/391 - 6s - loss: 2.0935 - accuracy: 0.2208 - val_loss: 2.1161 - val_accurac
y: 0.2396
Epoch 35/200
391/391 - 7s - loss: 2.1156 - accuracy: 0.2147 - val_loss: 2.1095 - val_accurac
y: 0.2286
Epoch 36/200
391/391 - 7s - loss: 2.0873 - accuracy: 0.2256 - val_loss: 2.1595 - val_accurac
y: 0.2328
Epoch 37/200
391/391 - 7s - loss: 2.1173 - accuracy: 0.2080 - val_loss: 2.0693 - val_accurac
y: 0.2626
Epoch 38/200
391/391 - 7s - loss: 2.1220 - accuracy: 0.2043 - val_loss: 2.1069 - val_accurac
y: 0.2245
Epoch 39/200
391/391 - 7s - loss: 2.1104 - accuracy: 0.2118 - val_loss: 2.1298 - val_accurac
y: 0.2030
Epoch 40/200
```

```
391/391 - 7s - loss: 2.0980 - accuracy: 0.2171 - val_loss: 2.0668 - val_accurac
y: 0.2473
Epoch 41/200
391/391 - 6s - loss: 2.0932 - accuracy: 0.2204 - val_loss: 2.1384 - val_accurac
y: 0.2096
Epoch 42/200
391/391 - 7s - loss: 2.0930 - accuracy: 0.2220 - val_loss: 2.0761 - val_accurac
y: 0.2474
Epoch 43/200
391/391 - 7s - loss: 2.0822 - accuracy: 0.2266 - val_loss: 2.1290 - val_accurac
y: 0.2147
Epoch 44/200
391/391 - 7s - loss: 2.0868 - accuracy: 0.2229 - val_loss: 2.0546 - val_accurac
y: 0.2475
Epoch 45/200
391/391 - 6s - loss: 2.0889 - accuracy: 0.2258 - val_loss: 2.0772 - val_accurac
y: 0.2574
Epoch 46/200
391/391 - 7s - loss: 2.0876 - accuracy: 0.2259 - val_loss: 2.0879 - val_accurac
y: 0.2377
Epoch 47/200
391/391 - 7s - loss: 2.1138 - accuracy: 0.2123 - val_loss: 2.0932 - val_accurac
y: 0.2488
Epoch 48/200
391/391 - 7s - loss: 2.0867 - accuracy: 0.2267 - val_loss: 2.0736 - val_accurac
y: 0.2368
Epoch 49/200
391/391 - 6s - loss: 2.1034 - accuracy: 0.2122 - val_loss: 2.1074 - val_accurac
y: 0.2210
Epoch 50/200
391/391 - 6s - loss: 2.1047 - accuracy: 0.2118 - val_loss: 2.1021 - val_accurac
y: 0.2115
Epoch 51/200
391/391 - 7s - loss: 2.1053 - accuracy: 0.2128 - val_loss: 2.1220 - val_accurac
y: 0.2216
Epoch 52/200
391/391 - 7s - loss: 2.1188 - accuracy: 0.2065 - val_loss: 2.1504 - val_accurac
y: 0.1913
Epoch 53/200
391/391 - 7s - loss: 2.1086 - accuracy: 0.2068 - val_loss: 2.1065 - val_accurac
y: 0.2303
Epoch 54/200
391/391 - 7s - loss: 2.1251 - accuracy: 0.2044 - val_loss: 2.1223 - val_accurac
y: 0.2131
Epoch 55/200
391/391 - 7s - loss: 2.1167 - accuracy: 0.2016 - val_loss: 2.1092 - val_accurac
y: 0.2082
Epoch 56/200
391/391 - 7s - loss: 2.1147 - accuracy: 0.2041 - val_loss: 2.1202 - val_accurac
y: 0.2118
Epoch 57/200
391/391 - 7s - loss: 2.1056 - accuracy: 0.2104 - val_loss: 2.1151 - val_accurac
y: 0.2278
Epoch 58/200
391/391 - 6s - loss: 2.1074 - accuracy: 0.2103 - val_loss: 2.1125 - val_accurac
y: 0.2047
Epoch 59/200
391/391 - 7s - loss: 2.1100 - accuracy: 0.2095 - val_loss: 2.1125 - val_accurac
y: 0.2353
```

```
Epoch 60/200
391/391 - 7s - loss: 2.1059 - accuracy: 0.2111 - val_loss: 2.1238 - val_accurac
y: 0.2266
Epoch 61/200
391/391 - 7s - loss: 2.1066 - accuracy: 0.2106 - val_loss: 2.1092 - val_accurac
y: 0.2186
Epoch 62/200
391/391 - 6s - loss: 2.1029 - accuracy: 0.2111 - val_loss: 2.0981 - val_accurac
y: 0.2069
Epoch 63/200
391/391 - 7s - loss: 2.0989 - accuracy: 0.2152 - val_loss: 2.0955 - val_accurac
y: 0.2210
Epoch 64/200
391/391 - 7s - loss: 2.1055 - accuracy: 0.2094 - val_loss: 2.0677 - val_accurac
y: 0.2294
Epoch 65/200
391/391 - 7s - loss: 2.1019 - accuracy: 0.2128 - val_loss: 2.0923 - val_accurac
y: 0.2515
Epoch 66/200
391/391 - 6s - loss: 2.1033 - accuracy: 0.2128 - val_loss: 2.0932 - val_accurac
y: 0.2162
Epoch 67/200
391/391 - 7s - loss: 2.1067 - accuracy: 0.2071 - val_loss: 2.1056 - val_accurac
y: 0.2276
Epoch 68/200
391/391 - 7s - loss: 2.1062 - accuracy: 0.2097 - val_loss: 2.1130 - val_accurac
y: 0.2293
Epoch 69/200
391/391 - 7s - loss: 2.0993 - accuracy: 0.2135 - val_loss: 2.0794 - val_accurac
y: 0.2297
Epoch 70/200
391/391 - 6s - loss: 2.0982 - accuracy: 0.2126 - val_loss: 2.0690 - val_accurac
y: 0.2289
Epoch 71/200
391/391 - 7s - loss: 2.1016 - accuracy: 0.2120 - val_loss: 2.1520 - val_accurac
y: 0.1911
Epoch 72/200
391/391 - 7s - loss: 2.1074 - accuracy: 0.2117 - val_loss: 2.1004 - val_accurac
y: 0.2109
Epoch 73/200
391/391 - 6s - loss: 2.1013 - accuracy: 0.2160 - val_loss: 2.0965 - val_accurac
y: 0.2258
Epoch 74/200
391/391 - 7s - loss: 2.1048 - accuracy: 0.2141 - val_loss: 2.0958 - val_accurac
y: 0.2273
Epoch 75/200
391/391 - 7s - loss: 2.1091 - accuracy: 0.2102 - val_loss: 2.1397 - val_accurac
y: 0.2004
Epoch 76/200
391/391 - 7s - loss: 2.1087 - accuracy: 0.2102 - val_loss: 2.0890 - val_accurac
y: 0.2236
Epoch 77/200
391/391 - 7s - loss: 2.1027 - accuracy: 0.2153 - val_loss: 2.0896 - val_accurac
y: 0.2348
Epoch 78/200
391/391 - 7s - loss: 2.1005 - accuracy: 0.2140 - val_loss: 2.0957 - val_accurac
y: 0.2293
Epoch 79/200
391/391 - 7s - loss: 2.1203 - accuracy: 0.2044 - val_loss: 2.0977 - val_accurac
```

```
y: 0.2150
Epoch 80/200
391/391 - 6s - loss: 2.1218 - accuracy: 0.2036 - val_loss: 2.1103 - val_accurac
y: 0.2016
Epoch 81/200
391/391 - 7s - loss: 2.1011 - accuracy: 0.2114 - val_loss: 2.0959 - val_accurac
y: 0.2384
Epoch 82/200
391/391 - 6s - loss: 2.1086 - accuracy: 0.2102 - val_loss: 2.1017 - val_accurac
y: 0.2290
Epoch 83/200
391/391 - 6s - loss: 2.1002 - accuracy: 0.2151 - val_loss: 2.1091 - val_accurac
y: 0.2031
Epoch 84/200
391/391 - 6s - loss: 2.0969 - accuracy: 0.2154 - val_loss: 2.0767 - val_accurac
y: 0.2290
Epoch 85/200
391/391 - 7s - loss: 2.0952 - accuracy: 0.2161 - val_loss: 2.1052 - val_accurac
y: 0.2403
Epoch 86/200
391/391 - 7s - loss: 2.1098 - accuracy: 0.2099 - val_loss: 2.1002 - val_accurac
y: 0.2295
Epoch 87/200
391/391 - 7s - loss: 2.0977 - accuracy: 0.2154 - val_loss: 2.1090 - val_accurac
y: 0.2262
Epoch 88/200
391/391 - 6s - loss: 2.0964 - accuracy: 0.2158 - val_loss: 2.1207 - val_accurac
y: 0.2055
Epoch 89/200
391/391 - 7s - loss: 2.1115 - accuracy: 0.2084 - val_loss: 2.1035 - val_accurac
y: 0.2413
Epoch 90/200
391/391 - 7s - loss: 2.1053 - accuracy: 0.2119 - val_loss: 2.1193 - val_accurac
y: 0.1952
Epoch 91/200
391/391 - 7s - loss: 2.1018 - accuracy: 0.2124 - val_loss: 2.1235 - val_accurac
y: 0.2253
Epoch 92/200
391/391 - 6s - loss: 2.1023 - accuracy: 0.2133 - val_loss: 2.0946 - val_accurac
y: 0.2134
Epoch 93/200
391/391 - 7s - loss: 2.1010 - accuracy: 0.2132 - val_loss: 2.0573 - val_accurac
y: 0.2417
Epoch 94/200
391/391 - 7s - loss: 2.1033 - accuracy: 0.2139 - val_loss: 2.1033 - val_accurac
y: 0.2290
Epoch 95/200
391/391 - 6s - loss: 2.1040 - accuracy: 0.2112 - val_loss: 2.0892 - val_accurac
y: 0.2226
Epoch 96/200
391/391 - 6s - loss: 2.1022 - accuracy: 0.2135 - val_loss: 2.0970 - val_accurac
y: 0.2250
Epoch 97/200
391/391 - 7s - loss: 2.1007 - accuracy: 0.2136 - val_loss: 2.0925 - val_accurac
y: 0.2303
Epoch 98/200
391/391 - 7s - loss: 2.0976 - accuracy: 0.2154 - val_loss: 2.1102 - val_accurac
y: 0.2343
Epoch 99/200
```

```
391/391 - 7s - loss: 2.1189 - accuracy: 0.2052 - val_loss: 2.1018 - val_accurac
y: 0.2418
Epoch 100/200
391/391 - 6s - loss: 2.1031 - accuracy: 0.2090 - val_loss: 2.1020 - val_accurac
y: 0.2123
Epoch 101/200
391/391 - 7s - loss: 2.1166 - accuracy: 0.2071 - val_loss: 2.1028 - val_accurac
y: 0.2204
Epoch 102/200
391/391 - 7s - loss: 2.1213 - accuracy: 0.2017 - val_loss: 2.1247 - val_accurac
y: 0.2129
Epoch 103/200
391/391 - 7s - loss: 2.1217 - accuracy: 0.2045 - val_loss: 2.1243 - val_accurac
y: 0.2216
Epoch 104/200
391/391 - 6s - loss: 2.1262 - accuracy: 0.2001 - val_loss: 2.1221 - val_accurac
y: 0.2002
Epoch 105/200
391/391 - 7s - loss: 2.1149 - accuracy: 0.2028 - val_loss: 2.1251 - val_accurac
y: 0.2210
Epoch 106/200
391/391 - 7s - loss: 2.1167 - accuracy: 0.2036 - val_loss: 2.1158 - val_accurac
y: 0.2136
Epoch 107/200
391/391 - 6s - loss: 2.1176 - accuracy: 0.2045 - val_loss: 2.1229 - val_accurac
y: 0.2226
Epoch 108/200
391/391 - 7s - loss: 2.1197 - accuracy: 0.2022 - val_loss: 2.0965 - val_accurac
y: 0.2303
Epoch 109/200
391/391 - 7s - loss: 2.1119 - accuracy: 0.2044 - val_loss: 2.1249 - val_accurac
y: 0.2246
Epoch 110/200
391/391 - 7s - loss: 2.1256 - accuracy: 0.1988 - val_loss: 2.0969 - val_accurac
y: 0.1984
Epoch 111/200
391/391 - 7s - loss: 2.1228 - accuracy: 0.2024 - val_loss: 2.0858 - val_accurac
y: 0.2109
Epoch 112/200
391/391 - 6s - loss: 2.1260 - accuracy: 0.1992 - val_loss: 2.1326 - val_accurac
y: 0.2197
Epoch 113/200
391/391 - 7s - loss: 2.1190 - accuracy: 0.2022 - val_loss: 2.1390 - val_accurac
y: 0.2202
Epoch 114/200
391/391 - 7s - loss: 2.1140 - accuracy: 0.2043 - val_loss: 2.1110 - val_accurac
y: 0.2198
Epoch 115/200
391/391 - 7s - loss: 2.1196 - accuracy: 0.2007 - val_loss: 2.1259 - val_accurac
y: 0.2226
Epoch 116/200
391/391 - 6s - loss: 2.1170 - accuracy: 0.2030 - val_loss: 2.1481 - val_accurac
y: 0.1951
Epoch 117/200
391/391 - 7s - loss: 2.1166 - accuracy: 0.2040 - val_loss: 2.1382 - val_accurac
y: 0.2103
Epoch 118/200
391/391 - 7s - loss: 2.1152 - accuracy: 0.2038 - val_loss: 2.1100 - val_accurac
y: 0.2068
```

```
Epoch 119/200
391/391 - 6s - loss: 2.1198 - accuracy: 0.2040 - val_loss: 2.1443 - val_accurac
y: 0.2027
Epoch 120/200
391/391 - 7s - loss: 2.1270 - accuracy: 0.1990 - val_loss: 2.1077 - val_accurac
y: 0.2010
Epoch 121/200
391/391 - 7s - loss: 2.1146 - accuracy: 0.2065 - val_loss: 2.1454 - val_accurac
y: 0.1914
Epoch 122/200
391/391 - 7s - loss: 2.1142 - accuracy: 0.2055 - val_loss: 2.1196 - val_accurac
y: 0.2277
Epoch 123/200
391/391 - 6s - loss: 2.1190 - accuracy: 0.2025 - val_loss: 2.1288 - val_accurac
y: 0.1949
Epoch 124/200
391/391 - 7s - loss: 2.1144 - accuracy: 0.2032 - val_loss: 2.1258 - val_accurac
y: 0.2042
Epoch 125/200
391/391 - 7s - loss: 2.1183 - accuracy: 0.2005 - val_loss: 2.1095 - val_accurac
y: 0.2072
Epoch 126/200
391/391 - 7s - loss: 2.1168 - accuracy: 0.2013 - val_loss: 2.1266 - val_accurac
y: 0.2021
Epoch 127/200
391/391 - 6s - loss: 2.1135 - accuracy: 0.2049 - val_loss: 2.1072 - val_accurac
y: 0.2191
Epoch 128/200
391/391 - 7s - loss: 2.1170 - accuracy: 0.2046 - val_loss: 2.1273 - val_accurac
y: 0.2161
Epoch 129/200
391/391 - 7s - loss: 2.1215 - accuracy: 0.2012 - val_loss: 2.0962 - val_accurac
y: 0.2235
Epoch 130/200
391/391 - 7s - loss: 2.1145 - accuracy: 0.2059 - val_loss: 2.1262 - val_accurac
y: 0.1957
Epoch 131/200
391/391 - 7s - loss: 2.1117 - accuracy: 0.2048 - val_loss: 2.1082 - val_accurac
y: 0.2140
Epoch 132/200
391/391 - 7s - loss: 2.1167 - accuracy: 0.2029 - val_loss: 2.1096 - val_accurac
y: 0.2027
Epoch 133/200
391/391 - 7s - loss: 2.1156 - accuracy: 0.2074 - val_loss: 2.0853 - val_accurac
y: 0.2494
Epoch 134/200
391/391 - 6s - loss: 2.1157 - accuracy: 0.2042 - val_loss: 2.1101 - val_accurac
y: 0.2159
Epoch 135/200
391/391 - 6s - loss: 2.1179 - accuracy: 0.2029 - val_loss: 2.1029 - val_accurac
y: 0.2224
Epoch 136/200
391/391 - 5s - loss: 2.1187 - accuracy: 0.2025 - val_loss: 2.1242 - val_accurac
y: 0.2034
Epoch 137/200
391/391 - 8s - loss: 2.1128 - accuracy: 0.2052 - val_loss: 2.1093 - val_accurac
y: 0.2199
Epoch 138/200
391/391 - 4s - loss: 2.1175 - accuracy: 0.2051 - val_loss: 2.1312 - val_accurac
```

```
y: 0.2128
Epoch 139/200
391/391 - 4s - loss: 2.1119 - accuracy: 0.2048 - val_loss: 2.1137 - val_accurac
y: 0.2071
Epoch 140/200
391/391 - 4s - loss: 2.1218 - accuracy: 0.2017 - val_loss: 2.1009 - val_accurac
y: 0.2204
Epoch 141/200
391/391 - 4s - loss: 2.1162 - accuracy: 0.2030 - val_loss: 2.0965 - val_accurac
y: 0.2175
Epoch 142/200
391/391 - 4s - loss: 2.1363 - accuracy: 0.1992 - val_loss: 2.1321 - val_accurac
y: 0.1969
Epoch 143/200
391/391 - 4s - loss: 2.1316 - accuracy: 0.1946 - val_loss: 2.1239 - val_accurac
y: 0.2025
Epoch 144/200
391/391 - 4s - loss: 2.1206 - accuracy: 0.2013 - val_loss: 2.1242 - val_accurac
y: 0.2147
Epoch 145/200
391/391 - 4s - loss: 2.1170 - accuracy: 0.2051 - val_loss: 2.1246 - val_accurac
y: 0.2215
Epoch 146/200
391/391 - 4s - loss: 2.1196 - accuracy: 0.2033 - val_loss: 2.1160 - val_accurac
y: 0.2064
Epoch 147/200
391/391 - 4s - loss: 2.1215 - accuracy: 0.2022 - val_loss: 2.1068 - val_accurac
y: 0.1928
Epoch 148/200
391/391 - 4s - loss: 2.1205 - accuracy: 0.2004 - val_loss: 2.1069 - val_accurac
y: 0.2164
Epoch 149/200
391/391 - 4s - loss: 2.1198 - accuracy: 0.2022 - val_loss: 2.1381 - val_accurac
y: 0.1911
Epoch 150/200
391/391 - 4s - loss: 2.1218 - accuracy: 0.2027 - val_loss: 2.1229 - val_accurac
y: 0.2170
Epoch 151/200
391/391 - 4s - loss: 2.1343 - accuracy: 0.1935 - val_loss: 2.1149 - val_accurac
y: 0.2172
Epoch 152/200
391/391 - 4s - loss: 2.1234 - accuracy: 0.2031 - val_loss: 2.0910 - val_accurac
y: 0.2300
Epoch 153/200
391/391 - 4s - loss: 2.1173 - accuracy: 0.2034 - val_loss: 2.1907 - val_accurac
y: 0.1808
Epoch 154/200
391/391 - 4s - loss: 2.1195 - accuracy: 0.2027 - val_loss: 2.0993 - val_accurac
y: 0.2098
Epoch 155/200
391/391 - 4s - loss: 2.1311 - accuracy: 0.2008 - val_loss: 2.1164 - val_accurac
y: 0.1972
Epoch 156/200
391/391 - 4s - loss: 2.1161 - accuracy: 0.2031 - val_loss: 2.1173 - val_accurac
y: 0.2310
Epoch 157/200
391/391 - 4s - loss: 2.1187 - accuracy: 0.2010 - val_loss: 2.1530 - val_accurac
y: 0.1854
Epoch 158/200
```

```
391/391 - 4s - loss: 2.1190 - accuracy: 0.2025 - val_loss: 2.1599 - val_accurac
y: 0.1974
Epoch 159/200
391/391 - 4s - loss: 2.1197 - accuracy: 0.2020 - val_loss: 2.0990 - val_accurac
y: 0.2146
Epoch 160/200
391/391 - 4s - loss: 2.1218 - accuracy: 0.2028 - val_loss: 2.1086 - val_accurac
y: 0.2260
Epoch 161/200
391/391 - 4s - loss: 2.1161 - accuracy: 0.2021 - val_loss: 2.1460 - val_accurac
y: 0.1971
Epoch 162/200
391/391 - 4s - loss: 2.1172 - accuracy: 0.2036 - val_loss: 2.1317 - val_accurac
y: 0.2079
Epoch 163/200
391/391 - 4s - loss: 2.1148 - accuracy: 0.2031 - val_loss: 2.0912 - val_accurac
y: 0.2267
Epoch 164/200
391/391 - 4s - loss: 2.1173 - accuracy: 0.2014 - val_loss: 2.0910 - val_accurac
y: 0.2423
Epoch 165/200
391/391 - 4s - loss: 2.1190 - accuracy: 0.2028 - val_loss: 2.1546 - val_accurac
y: 0.2028
Epoch 166/200
391/391 - 4s - loss: 2.1184 - accuracy: 0.2065 - val_loss: 2.1449 - val_accurac
y: 0.2001
Epoch 167/200
391/391 - 4s - loss: 2.1179 - accuracy: 0.2039 - val_loss: 2.1492 - val_accurac
y: 0.1862
Epoch 168/200
391/391 - 4s - loss: 2.1179 - accuracy: 0.2012 - val_loss: 2.1151 - val_accurac
y: 0.2023
Epoch 169/200
391/391 - 4s - loss: 2.1192 - accuracy: 0.2047 - val_loss: 2.1077 - val_accurac
y: 0.2234
Epoch 170/200
391/391 - 4s - loss: 2.1218 - accuracy: 0.2003 - val_loss: 2.1414 - val_accurac
y: 0.2210
Epoch 171/200
391/391 - 4s - loss: 2.1127 - accuracy: 0.2063 - val_loss: 2.1157 - val_accurac
y: 0.2155
Epoch 172/200
391/391 - 4s - loss: 2.1207 - accuracy: 0.2010 - val_loss: 2.1257 - val_accurac
y: 0.2014
Epoch 173/200
391/391 - 4s - loss: 2.1182 - accuracy: 0.2030 - val_loss: 2.1017 - val_accurac
y: 0.2176
Epoch 174/200
391/391 - 4s - loss: 2.1206 - accuracy: 0.2007 - val_loss: 2.1171 - val_accurac
y: 0.1994
Epoch 175/200
391/391 - 4s - loss: 2.1201 - accuracy: 0.2024 - val_loss: 2.1201 - val_accurac
y: 0.2085
Epoch 176/200
391/391 - 4s - loss: 2.1176 - accuracy: 0.1997 - val_loss: 2.0767 - val_accurac
y: 0.2300
Epoch 177/200
391/391 - 4s - loss: 2.1198 - accuracy: 0.2014 - val_loss: 2.1187 - val_accurac
y: 0.2064
```

```
Epoch 178/200
391/391 - 4s - loss: 2.1233 - accuracy: 0.2013 - val_loss: 2.1831 - val_accurac
y: 0.1796
Epoch 179/200
391/391 - 4s - loss: 2.1177 - accuracy: 0.2045 - val_loss: 2.1087 - val_accurac
y: 0.2231
Epoch 180/200
391/391 - 4s - loss: 2.1165 - accuracy: 0.2048 - val_loss: 2.1202 - val_accurac
y: 0.2401
Epoch 181/200
391/391 - 4s - loss: 2.1138 - accuracy: 0.2065 - val_loss: 2.1078 - val_accurac
y: 0.2205
Epoch 182/200
391/391 - 4s - loss: 2.1201 - accuracy: 0.2014 - val_loss: 2.1322 - val_accurac
y: 0.2089
Epoch 183/200
391/391 - 4s - loss: 2.1163 - accuracy: 0.2050 - val_loss: 2.1355 - val_accurac
y: 0.1917
Epoch 184/200
391/391 - 4s - loss: 2.1156 - accuracy: 0.2069 - val_loss: 2.1333 - val_accurac
y: 0.2173
Epoch 185/200
391/391 - 4s - loss: 2.1125 - accuracy: 0.2047 - val_loss: 2.0780 - val_accurac
y: 0.2396
Epoch 186/200
391/391 - 4s - loss: 2.1209 - accuracy: 0.2014 - val_loss: 2.1167 - val_accurac
y: 0.2199
Epoch 187/200
391/391 - 4s - loss: 2.1168 - accuracy: 0.2053 - val_loss: 2.1024 - val_accurac
y: 0.2436
Epoch 188/200
391/391 - 4s - loss: 2.1166 - accuracy: 0.2032 - val_loss: 2.1168 - val_accurac
y: 0.1956
Epoch 189/200
391/391 - 4s - loss: 2.1294 - accuracy: 0.2010 - val_loss: 2.1165 - val_accurac
y: 0.2004
Epoch 190/200
391/391 - 4s - loss: 2.1165 - accuracy: 0.2031 - val_loss: 2.1084 - val_accurac
y: 0.2198
Epoch 191/200
391/391 - 4s - loss: 2.1175 - accuracy: 0.2025 - val_loss: 2.1089 - val_accurac
y: 0.2007
Epoch 192/200
391/391 - 4s - loss: 2.1151 - accuracy: 0.2023 - val_loss: 2.1618 - val_accurac
y: 0.1863
Epoch 193/200
391/391 - 4s - loss: 2.1174 - accuracy: 0.2028 - val_loss: 2.1310 - val_accurac
y: 0.2112
Epoch 194/200
391/391 - 4s - loss: 2.1154 - accuracy: 0.2040 - val_loss: 2.1317 - val_accurac
y: 0.2135
Epoch 195/200
391/391 - 4s - loss: 2.1192 - accuracy: 0.2021 - val_loss: 2.1090 - val_accurac
y: 0.2069
Epoch 196/200
391/391 - 4s - loss: 2.1180 - accuracy: 0.2010 - val_loss: 2.1617 - val_accurac
y: 0.1881
Epoch 197/200
391/391 - 4s - loss: 2.1325 - accuracy: 0.1960 - val_loss: 2.1203 - val_accurac
```

```
y: 0.2043
Epoch 198/200
391/391 - 4s - loss: 2.1162 - accuracy: 0.2039 - val_loss: 2.1066 - val_accurac
y: 0.2235
Epoch 199/200
391/391 - 4s - loss: 2.1282 - accuracy: 0.1987 - val_loss: 2.1252 - val_accurac
y: 0.1999
Epoch 200/200
391/391 - 6s - loss: 2.1243 - accuracy: 0.2009 - val_loss: 2.1113 - val_accurac
y: 0.2138
```

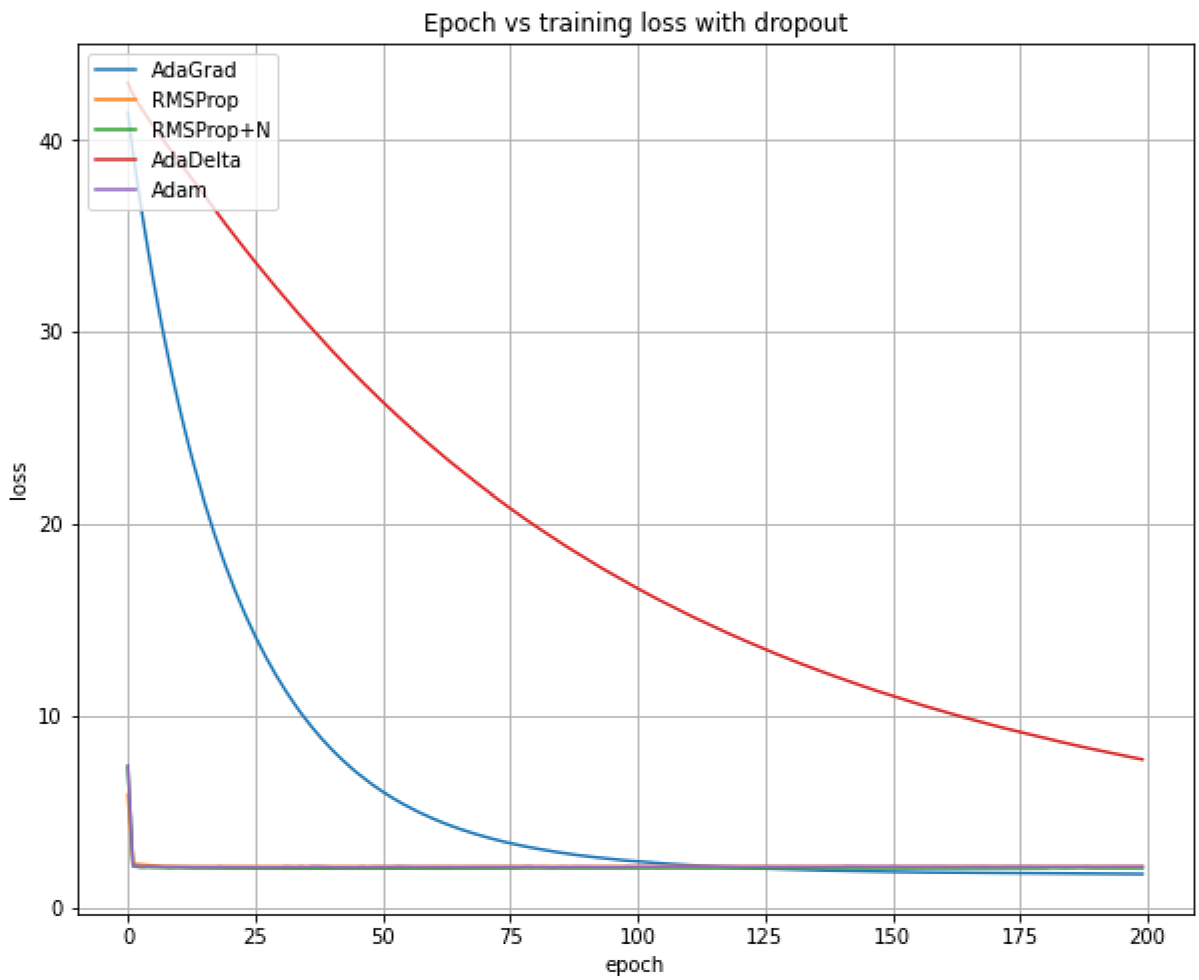
Lowest training loss

In [10]:
```python
print("lowest training loss")
print("AdaGrad: {}".format(min(history_adagrad_dropout.history['loss'])))
print("RMSProp: {}".format(min(history_rmsprop_dropout.history['loss'])))
print("RMSProp+N: {}".format(min(history_nesterov_dropout.history['loss'])))
print("AdaDelta: {}".format(min(history_adadelta_dropout.history['loss'])))
print("Adam: {}".format(min(history_adam_dropout.history['loss'])))
```

```
lowest training loss
AdaGrad: 1.7369143962860107
RMSProp: 2.115575075149536
RMSProp+N: 2.028042793273926
AdaDelta: 7.716529846191406
Adam: 2.0822479724884033
```

In [12]:

```
from matplotlib import pyplot as plt
#Plotting
fig = plt.figure(figsize=(10,8))
plt.plot(history_adagrad_dropout.history['loss'],label='AdaGrad')
plt.plot(history_rmsprop_dropout.history['loss'],label='RMSProp')
plt.plot(history_nesterov_dropout.history['loss'],label='RMSProp+N')
plt.plot(history_adadelta_dropout.history['loss'],label='AdaDelta')
plt.plot(history_adam_dropout.history['loss'],label='Adam')
plt.title('Epoch vs training loss with dropout')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(loc='upper left')
plt.grid()
plt.show()
```



Part 4

Compare test accuracy of the trained model for all the five methods from part 2 and part 3. Note that to calculate test accuracy of the model trained using dropout you need to appropriately scale the weights (by the dropout probability). (4)

Code

In [27]:
```
models_no_dropout=[model_adagrad,model_rmsprop,model_nesterov,model_adadelta,mode
l_adam]
models_dropout=[model_adagrad_dropout,model_rmsprop_dropout,model_nesterov_dropou
t,model_adadelta_dropout,model_adam_dropout]
optimizer_names=['Adagrad','RMSProp','RMSProp+Nesterov','Adadelta','Adam']


print("Test accuracy - models without dropout")
for i in range(len(models_no_dropout)):
    print("Test Accuracy - {}: {}".format(optimizer_names[i],models_no_dropout[i]
.evaluate(X_test, y_test)[1]))

print("Test accuracy - models with dropout")
for i in range(len(models_dropout)):
    print("Test Accuracy - {}: {}".format(optimizer_names[i],models_dropout[i].ev
aluate(X_test, y_test)[1]))
```

```
Test accuracy - models without dropout
313/313 [==============================] - 1s 2ms/step - loss: 1.6799 - accurac
y: 0.5031
Test Accuracy - Adagrad: 0.5030999779701233
313/313 [==============================] - 1s 2ms/step - loss: 1.9086 - accurac
y: 0.3608
Test Accuracy - RMSProp: 0.36079999804496765
313/313 [==============================] - 1s 2ms/step - loss: 1.5688 - accurac
y: 0.4755
Test Accuracy - RMSProp+Nesterov: 0.4754999876022339
313/313 [==============================] - 1s 2ms/step - loss: 7.1394 - accurac
y: 0.4537
Test Accuracy - Adadelta: 0.4537000060081482
313/313 [==============================] - 1s 2ms/step - loss: 1.7330 - accurac
y: 0.4200
Test Accuracy - Adam: 0.41999998688697815
Test accuracy - models with dropout
313/313 [==============================] - 1s 2ms/step - loss: 1.6934 - accurac
y: 0.4811
Test Accuracy - Adagrad: 0.4810999929904938
313/313 [==============================] - 1s 2ms/step - loss: 2.0576 - accurac
y: 0.2864
Test Accuracy - RMSProp: 0.2863999903202057
313/313 [==============================] - 1s 2ms/step - loss: 2.0737 - accurac
y: 0.2396
Test Accuracy - RMSProp+Nesterov: 0.23960000276565552
313/313 [==============================] - 1s 2ms/step - loss: 7.6006 - accurac
y: 0.3985
Test Accuracy - Adadelta: 0.398499995470047
313/313 [==============================] - 1s 2ms/step - loss: 2.1113 - accurac
y: 0.2138
Test Accuracy - Adam: 0.21379999816417694
```

Problem 5 - Convolutional Neural Networks Architectures

Part 1

Calculate the number of parameters in Alexnet. You will have to show calculations for each layer and then sum it to obtain the total number of parameters in Alexnet. When calculating you will need to account for all the filters (size, strides, padding) at each layer. Look at Sec. 3.5 and Figure 2 in Alexnet paper (see reference). Points will only be given when explicit calculations are shown for each layer. (4)

Written Answer:

Equations:

- Parameters = (size of kernels^2 # *of channels in input image* number of kernels) + number of kernels
- Output of ConvLayer = (input - size of kernels + 2*padding)/stride + 1
- Output of MaxPool = (input - pool size)/stride + 1

Calculation

- Output = $\frac{227-11+2(0)}{4} + 1 = 55$
- CONV1: $(11 * 11) * (3) * (96) + 96 = 34944$
  - size: 55 x 55 x 96
  - Output = $\frac{55-3}{2} + 1 = 27$
- POOL1: 0 parameters;
  - size: 27 x 27 x 96
- Output = $\frac{27-5+2(2)}{1} + 1 = 27$
- CONV2: $(5 * 5) * (96) * (256) + 256 = 614656$
  - size: 27 x 27 x 256
- Output = $\frac{27-3}{2} + 1 = 13$
- POOL2: 0 parameters;
  - size: 13 x 13 x 256
- Output = $\frac{13-3+2(1)}{1} + 1 = 13$
- CONV3: $(3 * 3) * (256) * (384) + 384 = 885120$
  - size: 13 x 13 x 384
  - Output = $\frac{13-3+2(1)}{1} + 1 = 13$
- CONV4: $(3 * 3) * (384) * (384) + 384 = 1327488$
  - size: 13 x 13 x 384
- Output = $\frac{13-3+2(1)}{1} + 1 = 13$
- CONV5: $(3 * 3) * (384) * (256) + 256 = 884992$
  - size: 13 x 13 x 256
- Output = $\frac{13-3}{2} + 1 = 6$
- POOL5: 0 parameters;
  - size: 6 x 6 x 256
- FC1: $(6 * 6) * (256) * (4096) + 4096 = 37752832$
  - size = 4096 x 1
- FC2: $(4096) * (4096) + 4096 = 16781312$
  - size = 4096 x 1
- FC3: $(4096) * (1000) + 1000 = 4097000$
  - size = 4096 x 1

Total Number of parameters: $62, 378, 344$

Part 2

VGG (Simonyan et al.) has an extremely homogeneous architecture that only performs 3x3 convolutions with stride 1 and pad 1 and 2x2 max pooling with stride 2 (and no padding) from the beginning to the end. However VGGNet is very expensive to evaluate and uses a lot more memory and parameters. Refer to VGG19 architecture on page 3 in Table 1 of the paper by Simonyan et al. You need to complete Table 1 below for calculating activation units and parameters at each layer in VGG19 (without counting biases). Its been partially filled for you. (6)

Written Answer

Please refer to pdf file for detailed calculation.

Only include the missing entries

- CONV3-128: Memory: 1605632; Parameters: 73728
- CONV3-128: Memory: 1605632; Parameters: 147456
- CONV3-256: Memory: 802816; Parameters: 294912
- CONV3-256: Memory: 802816; Parameters: 589824
- CONV3-256: Memory: 802816; Parameters: 589824
- POOL2: Memory: 200704
- CONV3-512: Memory: 401408; Parameters: 1179648
- CONV3-512: Memory: 401408; Parameters: 2359296
- POOL2: Memory: 100353
- CONV3-512: Memory: 100352; Parameters: 2359296
- CONV3-512: Memory: 100352; Parameters: 2359296
- CONV3-512: Memory: 100352; Parameters: 2359296
- CONV3-512: Memory: 100352; Parameters: 2359296
- POOL2: Memory: 25088
- FC: Parameters: 102760488
- FC: Parameters: 4096000

Total: Memory: about $16.5M$; Parameters: about $140M$:

Part 3

VGG architectures have smaller filters but deeper networks compared to Alexnet (3x3 compared to 11x11 or 5x5). Show that a stack of N convolution layers each of filter size F × F has the same receptive field as one convolution layer with filter of size (NF − N + 1) × (NF − N + 1). Use this to calculate the receptive field of 3 filters of size 5x5. (3)

Written Answer

Architecture with a stack of N convolution layer and filter size FxF:

- Each Layer
  - $L_1^K = L - F + 1,$
    - L: input shape
    - K kernel
- N Layers:
  - $L_n^K = L - N(F + 1)$

Architecture with one convolution layer and filter of size (NF-N+1)x(NF-N+1),

- $L - (NF - N + 1) + 1 = L - N(F + 1)$

Part 4

The original Googlenet paper (Szegedy et al.) proposes two architectures for Inception module, shown in Figure 2 on page 5 of the paper, referred to as naive and dimensionality reduction respectively.

Written Answer

(a) What is the general idea behind designing an inception module (parallel convolutional filters of different sizes with a pooling followed by concatenation) in a convolutional neural network ? (2)

- The general idea behind designing an inception module is the efficient use of computing resource with minimal increase in computation load by using the matrix-multiplication routine to convert sparse matrices from kernels into a denser format.

(b) Assuming the input to inception module (referred to as ”previous layer” in Figure 2 of the pa- per) has size 32x32x256, calculate the output size after filter concatenation for the naive and dimensionality reduction inception architectures with number of filters given in Figure 1. (3)

- Naive inception architecture: $32 * 32 * (128 + 192 + 96 + 256) = 32 * 32 * 672$
- Dimensionality reduction architecture: $32 * 32 * (128 + 192 + 96 + 64) = 32 * 32 * 480$

(c) Next calculate the total number of convolutional operations for each of the two inception archi- tecture again assuming the input to the module has dimensions 32x32x256 and number of filters given in Figure 1. (3)

Naive inception architecture:

```
- Conv1: 32*32*1*256*128=33554432
- Conv2: 32*32*9*256*128=301989888
- Conv3: 32*32*25*256*128=838860800
- Total: 1174405120
```

Dimensionality reduction architecture:

```
- Conv1: 32*32*256*128 + 32*32*256*128 + 32*32*256*64=92274688
- Conv3: 32*32*9*128*192=226492416
- Conv5: 32*32*25*32*96=78643200
- Total: 397410304
```

(d) Based on the calculations in part (c) explain the problem with naive architecture and how dimensionality reduction architecture helps (Hint: compare computational complexity). How much is the computational saving ? (2+2)

- Dimensionality reduction architecture helps reduce computational complexity by having smaller size of outputs for each inception module.
- The naive architecture is 2.96 times more computational complexity than Dimensionality reduction architecture.