main

March 23, 2022

[2]:
```
# Import required packages
!pip install -U tensorflow-addons
!pip install -U opencv-python
import numpy as np
import cv2
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow_addons as tfa
from IPython.display import Image
```

```
Requirement already satisfied: tensorflow-addons in
c:\users\aroni\anaconda3\lib\site-packages (0.16.1)
Requirement already satisfied: typeguard>=2.7 in
c:\users\aroni\anaconda3\lib\site-packages (from tensorflow-addons) (2.13.3)

WARNING: You are using pip version 22.0.2; however, version 22.0.4 is available.
You should consider upgrading via the 'c:\users\aroni\anaconda3\python.exe -m
pip install --upgrade pip' command.

Requirement already satisfied: opencv-python in
c:\users\aroni\anaconda3\lib\site-packages (4.5.5.64)
Requirement already satisfied: numpy>=1.14.5 in
c:\users\aroni\anaconda3\lib\site-packages (from opencv-python) (1.20.1)

WARNING: You are using pip version 22.0.2; however, version 22.0.4 is available.
You should consider upgrading via the 'c:\users\aroni\anaconda3\python.exe -m
pip install --upgrade pip' command.
```

[2]:
```
tf.config.list_physical_devices('GPU')
```

```
2022-03-23 20:26:49.005932: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-03-23 20:26:49.017312: I
```

```
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-03-23 20:26:49.018040: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
```

[2]:
```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

0.1 1. Load the datasets

For the project, we provide a training set with 50000 images in the directory `../data/images/` with: - noisy labels for all images provided in `../data/noisy_label.csv`; - clean labels for the first 10000 images provided in `../data/clean_labels.csv`.

[3]:
```python
# [DO NOT MODIFY THIS CELL]

# load the images
n_img = 50000
n_noisy = 40000
n_clean_noisy = n_img - n_noisy
imgs = np.empty((n_img,32,32,3))
for i in range(n_img):
    img_fn = f'../data/images/{i+1:05d}.png'
    imgs[i,:,:,:]=cv2.cvtColor(cv2.imread(img_fn),cv2.COLOR_BGR2RGB)

# load the labels
clean_labels = np.genfromtxt('../data/clean_labels.csv', delimiter=',',␣
 ↪dtype="int8")
noisy_labels = np.genfromtxt('../data/noisy_labels.csv', delimiter=',',␣
 ↪dtype="int8")
```

[4]:
```python
clean_labels = np.genfromtxt('../data/clean_labels.csv', delimiter=',',␣
 ↪dtype="int8")
print(clean_labels)
```

```
[6 9 9 … 1 1 5]
```

For illustration, we present a small subset (of size 8) of the images with their clean and noisy labels in `clean_noisy_trainset`. You are encouraged to explore more characteristics of the label noises on the whole dataset.

[5]:
```python
# [DO NOT MODIFY THIS CELL]

fig = plt.figure()

ax1 = fig.add_subplot(2,4,1)
ax1.imshow(imgs[0]/255)
```

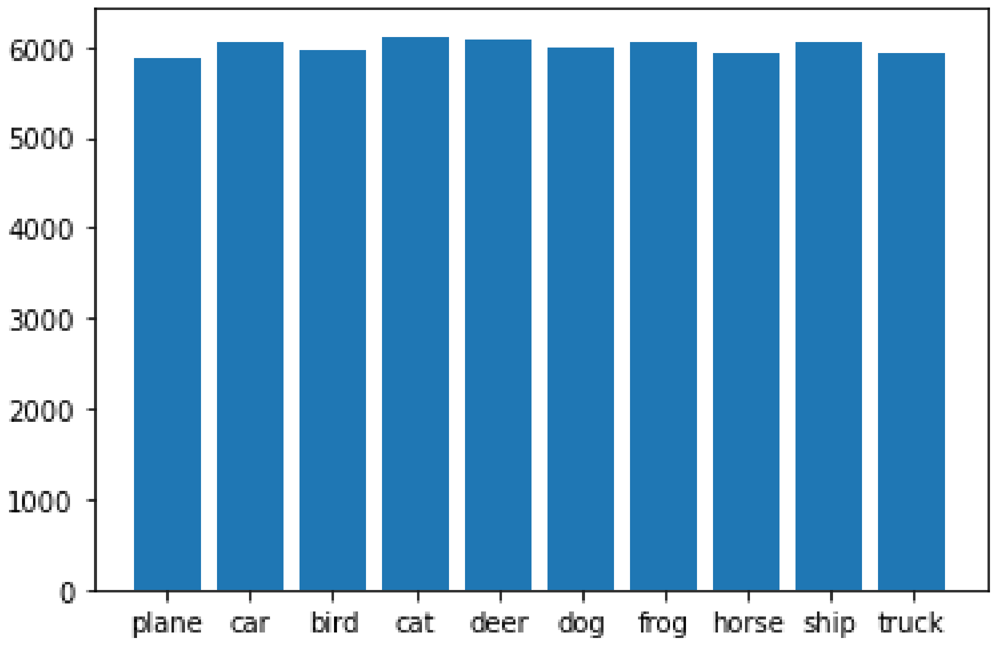
```
ax2 = fig.add_subplot(2,4,2)
ax2.imshow(imgs[1]/255)
ax3 = fig.add_subplot(2,4,3)
ax3.imshow(imgs[2]/255)
ax4 = fig.add_subplot(2,4,4)
ax4.imshow(imgs[3]/255)
ax1 = fig.add_subplot(2,4,5)
ax1.imshow(imgs[4]/255)
ax2 = fig.add_subplot(2,4,6)
ax2.imshow(imgs[5]/255)
ax3 = fig.add_subplot(2,4,7)
ax3.imshow(imgs[6]/255)
ax4 = fig.add_subplot(2,4,8)
ax4.imshow(imgs[7]/255)

# The class-label correspondence
classes = ('plane', 'car', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

# print clean labels
print('Clean labels:')
print(' '.join('%5s' % classes[clean_labels[j]] for j in range(8)))
# print noisy labels
print('Noisy labels:')
print(' '.join('%5s' % classes[noisy_labels[j]] for j in range(8)))
```

```
Clean labels:
 frog truck truck  deer   car   car  bird horse
Noisy labels:
  cat   dog truck  frog   dog  ship  bird  deer
```

Let's have a look at how the classes are distributed among the images. Indeed, we have to make sure that one class is not over represented to avoid overfitting over this class. We see from below that each class has somehow the same frequency of allocation among the images of our dataset. Note that this is the case for the noisy labels, but it may not be the case for the "cleaned" labels.

```
[6]: label_named = pd.Series([classes[index] for index in np.
      ↪concatenate([clean_labels,noisy_labels])])
     dict_class_frequency = {}
     for classe in classes:
         dict_class_frequency[classe] = label_named.str.count(classe).sum()
     plt.bar(list(dict_class_frequency),dict_class_frequency.values())
```

```
[6]: <BarContainer object of 10 artists>
```



1 2. The predictive model

We consider a baseline model directly on the noisy dataset without any label corrections. RGB histogram features are extracted to fit a logistic regression model.

1.0.1 2.1. Baseline Model

[7]:
```
# [DO NOT MODIFY THIS CELL]
# RGB histogram dataset construction
no_bins = 6
bins = np.linspace(0,255,no_bins) # the range of the rgb histogram
target_vec = np.empty(n_img)
feature_mtx = np.empty((n_img,3*(len(bins)-1)))
i = 0
for i in range(n_img):
    # The target vector consists of noisy labels
    target_vec[i] = noisy_labels[i]

    # Use the numbers of pixels in each bin for all three channels as the␣
 ↪features
    feature1 = np.histogram(imgs[i][:,:,0],bins=bins)[0]
    feature2 = np.histogram(imgs[i][:,:,1],bins=bins)[0]
    feature3 = np.histogram(imgs[i][:,:,2],bins=bins)[0]

    # Concatenate three features
    feature_mtx[i,] = np.concatenate((feature1, feature2, feature3), axis=None)
    i += 1
```

[8]:
```
# [DO NOT MODIFY THIS CELL]
# Train a logistic regression model
clf = LogisticRegression(random_state=0).fit(feature_mtx, target_vec)
```

For the convenience of evaluation, we write the following function `predictive_model` that does the label prediction. **For your predictive model, feel free to modify the function, but make sure the function takes an RGB image of numpy.array format with dimension $32 \times 32 \times 3$ as input, and returns one single label as output.**

[9]:
```
# [DO NOT MODIFY THIS CELL]
def baseline_model(image):
    '''
    This is the baseline predictive model that takes in the image and returns a␣
 ↪label prediction
    '''
    feature1 = np.histogram(image[:,:,0],bins=bins)[0]
    feature2 = np.histogram(image[:,:,1],bins=bins)[0]
    feature3 = np.histogram(image[:,:,2],bins=bins)[0]
    feature = np.concatenate((feature1, feature2, feature3), axis=None).
 ↪reshape(1,-1)
    return clf.predict(feature)
```

1.0.2 2.2. Model I

Before training our model, let's first prepare our data so our model can easily learn from it. First of all, we convert our image RGB value into `float32` and normalzing it to 0-1. Then we apply a label smoothing to our labels. Basically, the idea is to suggest that the labeling may be inaccurate and avoid overconfident prediction. This is a very good setting for us since we know our label are inacurate so we want our model to not be over confident over the prediction.

We then separate our data set into training set and test set. The last 45000 images with their noisy labels are used for training and the first 5000 images with their true label are used for testing.

```
[10]: labels = np.concatenate([clean_labels,noisy_labels[10000:]])
      def prep_pixels(train):
              # convert from integers to floats
              train_norm = train.astype('float32')
              # normalize to range 0-1
              train_norm = train_norm / 255.0
              # return normalized images
              return train_norm
      def smooth_labels(labels, factor=0.1):
              # smooth the labels
              labels *= (1 - factor)
              labels += (factor / labels.shape[1])
              # returned the smoothed labels
              return labels
      imgs_prep = prep_pixels(imgs)
      train = imgs_prep[5000:50000]
      test = imgs_prep[:5000]
      train_label = smooth_labels(keras.utils.to_categorical(noisy_labels[5000:
       ↪50000]))
      test_labels = smooth_labels(keras.utils.to_categorical(clean_labels[:5000]))
```

Hyperparameters of our models :

```
[11]: num_classes = 10
      input_shape = (32, 32, 3)
      learning_rate = 0.001
      weight_decay = 0.0001
      batch_size = 128
      num_epochs = 100
      image_size = 32  # We'll resize input images to this size

      positional_emb = True
      conv_layers = 2
      projection_dim = 128

      num_heads = 2
      transformer_units = [
          projection_dim,
          projection_dim,
      ]
      transformer_layers = 2
      stochastic_depth_rate = 0.1
```

Again to improve the training of our model and avoid overfitting, we use a very common technique that is data augmentation. In this case, for each images, we create a new image that was either cropped or flipped horizontally or rotated by a random angle or zoomed in by a random factor. As we are gonna see later, an other data augmentation technique can be very handfull in noisy label training which is know as mixup.

[12]:
```
data_augmentation = keras.Sequential(
    [
        layers.RandomCrop(image_size, image_size),
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(factor=0.02),
        layers.RandomZoom(
            height_factor=0.2, width_factor=0.2
        ),
    ],
    name="data_augmentation",
)
```

```
2022-03-23 20:27:08.393448: I tensorflow/core/platform/cpu_feature_guard.cc:151]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2022-03-23 20:27:08.394153: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-03-23 20:27:08.394998: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-03-23 20:27:08.395614: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-03-23 20:27:08.932069: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-03-23 20:27:08.932793: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
```

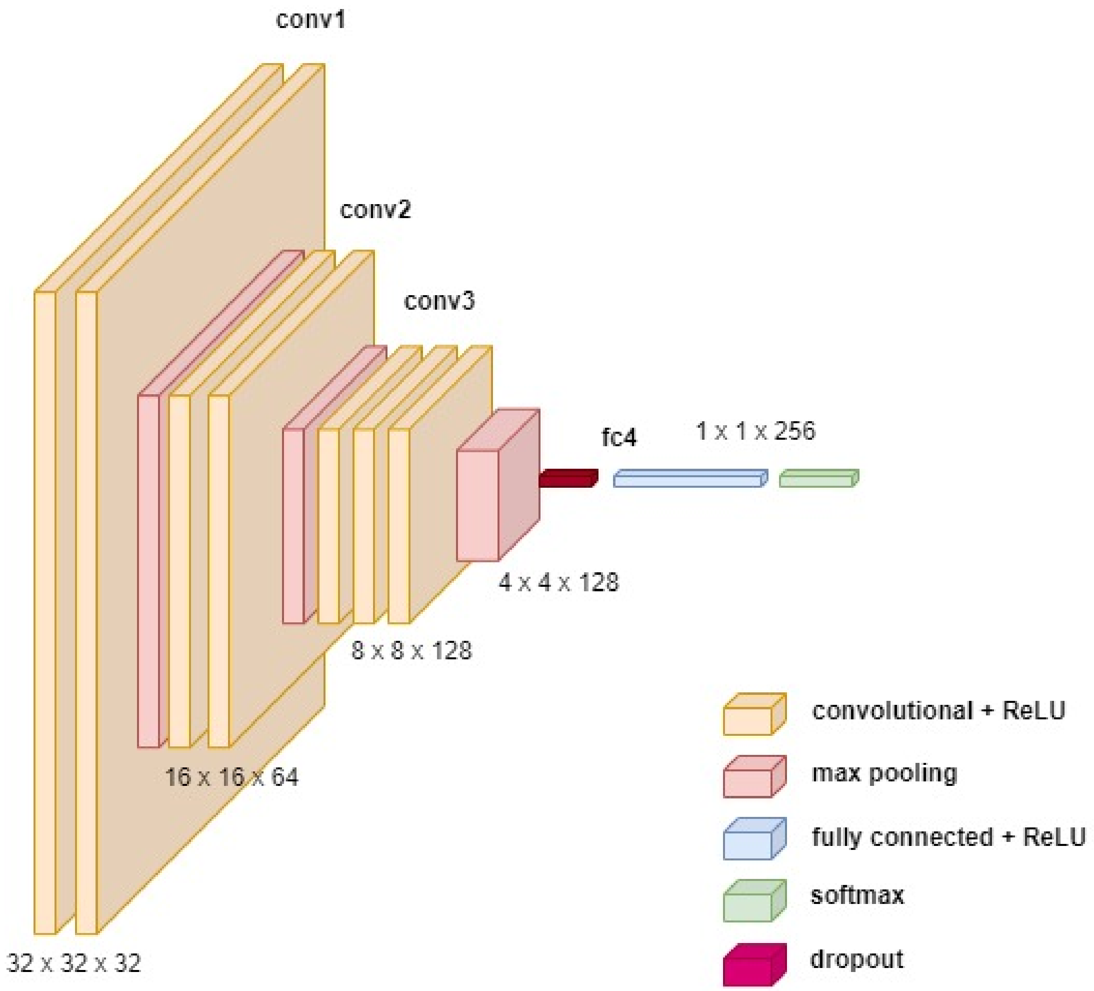
```
node, so returning NUMA node zero
2022-03-23 20:27:08.933447: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-03-23 20:27:08.934104: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1525] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 13821 MB memory:  -> device:
0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5
```

1.1 Implement the VGG classifier

The VGG classifier was our first model and appeared to be the most reliable for this task. Its architecture is shown in the figure below :

[3]:
```
Image(filename = "../figs/vgg16_xml.jpg")
```

[3]:



It is composed of a sequence of 3 Convolutional block each reducing the features map by 2 and increasing the filter by 2 as well (features map : {32,16,8} and filer : {32,64,128}). This Convolutional block are composed of 2 Convolutional layers with a kernel of (3,3) and a padding of 2 so the output has the same size as the input, and a Max Pooling with a kernel of (2,2). The output of this sequence of blocks is fed into a dropout layer with a dropout frequency of 0.5. Since we have a noisy data set, we have to regularize our model so it doesn't over-fit on the training set : this droupout layer forces our model to adapt itself and leverage over-fitting. The last 2 layers form a typical Dense Classification layer with a ReLU dense layer and a Softmax dense layer.

[13]:
```
def create_vgg_classifier(num_layer):
    inputs = layers.Input(shape=input_shape)
    # Augment data.
    vgg_encoded = data_augmentation(inputs)
    layer_input = image_size
    for _ in range(num_layer):
        vgg_encoded = layers.Conv2D(layer_input, (3, 3),
            activation='relu', kernel_initializer='he_uniform',
            padding='same', input_shape=(layer_input, layer_input, 3)
            )(vgg_encoded)
        vgg_encoded = layers.Conv2D(layer_input, (3, 3),
            activation='relu', kernel_initializer='he_uniform',
            padding='same', input_shape=(layer_input, layer_input, 3)
            )(vgg_encoded)
        vgg_encoded = layers.MaxPooling2D((2,2))(vgg_encoded)
        layer_input *=2
    representation = layers.Flatten()(vgg_encoded)
    representation = layers.Dropout(0.5)(representation)
    features = layers.
↪Dense(layer_input,activation="relu",kernel_initializer="he_uniform")(representation)
    logits = layers.Dense(num_classes)(features)
    # Create the Keras model.
    model = keras.Model(inputs=inputs, outputs=logits)
    return model
```

1.2 Implement the CCT model

This model is the second model that was tested against the VGG model and the baseline model. Since it is not our main model we won't go to far in the implementation details. Convolutional Compact Transformer model use the based framework of Vision Transformer model but instead of applying a linear token embedding in the beginning they use a Convolution Block. This allow to encode neighboring information in each token patch which is not possible with only a linear embedding. For those curious in understanding the implementation you can have a look at Escaping the Big Data Paradigm with Compact Transformers.

Implement MLP

[14]:
```
def mlp(x, hidden_units, dropout_rate):
    for units in hidden_units:
        x = layers.Dense(units, activation=tf.nn.gelu)(x)
        x = layers.Dropout(dropout_rate)(x)
    return x
```

Implement Convolutional tokenizer as a layer

[15]:
```
class Cliper(layers.Layer):
    def __init__(self):
        super(Cliper, self).__init__()

    def call(self,label):
        cliped_label = keras.backend.clip(label,0,1)
        return cliped_label
```

[16]:
```
class CCTTokenizer(layers.Layer):
    def __init__(
        self,
        kernel_size=3,
        stride=1,
        padding=1,
        pooling_kernel_size=3,
        pooling_stride=2,
        num_conv_layers=conv_layers,
        num_output_channels=[64, 128],
        positional_emb=positional_emb,
        **kwargs,
    ):
        super(CCTTokenizer, self).__init__(**kwargs)

        # This is our tokenizer.
        self.conv_model = keras.Sequential()
        for i in range(num_conv_layers):
            self.conv_model.add(
                layers.Conv2D(
                    num_output_channels[i],
                    kernel_size,
                    stride,
                    padding="valid",
                    use_bias=False,
                    activation="relu",
                    kernel_initializer="he_normal",
                )
            )
            self.conv_model.add(
                layers.Conv2D(
                    num_output_channels[i],
                    kernel_size,
                    stride,
                    padding="valid",
                    use_bias=False,
                    activation="relu",
                    kernel_initializer="he_normal",
                )
            )
            self.conv_model.add(
                layers.MaxPooling2D(pooling_kernel_size, pooling_stride, "same")
            )

        self.positional_emb = positional_emb

    def call(self, images):
        outputs = self.conv_model(images)
        # After passing the images through our mini-network the spatial␣
↪dimensions
        # are flattened to form sequences.
        reshaped = tf.reshape(
            outputs,
            (-1, tf.shape(outputs)[1] * tf.shape(outputs)[2], tf.
↪shape(outputs)[-1]),
        )
        return reshaped

    def positional_embedding(self, image_size):
        # Positional embeddings are optional in CCT. Here, we calculate
        # the number of sequences and initialize an `Embedding` layer to
        # compute the positional embeddings later.
        if self.positional_emb:
            dummy_inputs = tf.ones((1, image_size, image_size, 3))
            dummy_outputs = self.call(dummy_inputs)
            sequence_length = tf.shape(dummy_outputs)[1]
            projection_dim = tf.shape(dummy_outputs)[-1]

            embed_layer = layers.Embedding(
                input_dim=sequence_length, output_dim=projection_dim
            )
            return embed_layer, sequence_length
        else:
            return None
```

Stochastic depth for regularization

[17]:
```
# Referred from: github.com:rwightman/pytorch-image-models.
class StochasticDepth(layers.Layer):
    def __init__(self, drop_prop, **kwargs):
        super(StochasticDepth, self).__init__(**kwargs)
        self.drop_prob = drop_prop

    def call(self, x, training=None):
        if training:
            keep_prob = 1 - self.drop_prob
            shape = (tf.shape(x)[0],) + (1,) * (len(tf.shape(x)) - 1)
            random_tensor = keep_prob + tf.random.uniform(shape, 0, 1)
            random_tensor = tf.floor(random_tensor)
            return (x / keep_prob) * random_tensor
        return x
```

[18]:
```
def create_cct_classifier():
    inputs = layers.Input(input_shape)

    # Augment data.
    augmented = data_augmentation(inputs)

    # Encode patches.
    cct_tokenizer = CCTTokenizer()
    encoded_patches = cct_tokenizer(augmented)

    # Apply positional embedding.
    if positional_emb:
        pos_embed, seq_length = cct_tokenizer.positional_embedding(image_size)
        positions = tf.range(start=0, limit=seq_length, delta=1)
        position_embeddings = pos_embed(positions)
        encoded_patches += position_embeddings

    # Calculate Stochastic Depth probabilities.
    dpr = [x for x in np.linspace(0, stochastic_depth_rate, transformer_layers)]

    # Create multiple layers of the Transformer block.
    for i in range(transformer_layers):
        # Layer normalization 1.
        x1 = layers.LayerNormalization(epsilon=1e-5)(encoded_patches)

        # Create a multi-head attention layer.
        attention_output = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=projection_dim, dropout=0.1
        )(x1, x1)

        # Skip connection 1.
        attention_output = StochasticDepth(dpr[i])(attention_output)
        x2 = layers.Add()([attention_output, encoded_patches])

        # Layer normalization 2.
        x3 = layers.LayerNormalization(epsilon=1e-5)(x2)
        # MLP.
        x3 = mlp(x3, hidden_units=transformer_units, dropout_rate=0.1)

        # Skip connection 2.
        x3 = StochasticDepth(dpr[i])(x3)
        encoded_patches = layers.Add()([x3, x2])

    # Apply sequence pooling.
    representation = layers.LayerNormalization(epsilon=1e-5)(encoded_patches)
    attention_weights = tf.nn.softmax(layers.Dense(1)(representation), axis=1)
    weighted_representation = tf.matmul(
        attention_weights, representation, transpose_a=True
    )
    weighted_representation = tf.squeeze(weighted_representation, -2)
    weighted_representation = layers.Dropout(0.1)(weighted_representation)
    weighted_representation = layers.
↪Dense(256,activation="relu",kernel_initializer="he_uniform")(weighted_representation)
    # Classify outputs.
    logits = layers.Dense(num_classes)(weighted_representation)
    # Create the Keras model.
    model = keras.Model(inputs=inputs, outputs=logits)
    return model
```

1.3 Implement ResNet

This model will constitue our 3rd and 4th models that were tested. Again since it is not our main model we won't go to far in the implementation details. ResNet model (Deep Residual Learning for Image Recognition) is rather similar to the VGG model describes above. Instead of having 2 Convolution layers in each block, we have $2n$ Convolution layers and between each 2 layers we have residual (or skip connection). Each Convolution layers is also separated by a Batch Normalization layer. Also there is no Pooling layer between each convolution block. Instead to reduce the feature maps dimensions, we increase the stride of the first Convulotional layer to 2.

```
[19]: def create_resnet_classifier(n):
    inputs = layers.Input(shape=input_shape)
    # Augment data.
    augmented = data_augmentation(inputs)
    # First convolutional layer
    residual = layers.Conv2D(16, (3, 3),
            activation='relu', kernel_initializer='he_uniform',
            padding='same')(augmented)
    residual = layers.BatchNormalization(trainable=True)(residual)
    # Phase 1
    for _ in range(1,n+1):
        vgg_encoded = layers.Conv2D(16, (3, 3), kernel_initializer='he_uniform',
            padding='same')(residual)
        vgg_encoded = layers.BatchNormalization(trainable=True)(vgg_encoded)
        vgg_encoded = keras.activations.relu(vgg_encoded)
        vgg_encoded = layers.Conv2D(16, (3, 3), kernel_initializer='he_uniform',
            padding='same',activation='relu')(vgg_encoded)
        vgg_encoded = layers.BatchNormalization(trainable=True)(vgg_encoded)
        #vgg_encoded = layers.MaxPooling2D((2,2))(vgg_encoded)
        residual = layers.Add()([vgg_encoded,residual])
    # Phase 2
    for layer in range(1,n+1):
        if layer == 1 :
            vgg_encoded = layers.Conv2D(32, (3, 3),
                strides = 2, kernel_initializer='he_uniform',
                padding='same')(residual)
            vgg_encoded = layers.BatchNormalization(trainable=True)(vgg_encoded)
            vgg_encoded = keras.activations.relu(vgg_encoded)
            vgg_encoded = layers.Conv2D(32, (3, 3),activation="relu",␣
↪kernel_initializer='he_uniform',
                padding='same')(vgg_encoded)
            vgg_encoded = layers.BatchNormalization(trainable=True)(vgg_encoded)
            residual = layers.Conv2D(32, (3, 3),
                strides = 2, kernel_initializer='he_uniform',
                padding='same')(residual)
            residual = layers.Add()([vgg_encoded,residual])
        else :
            vgg_encoded = layers.Conv2D(32, (3, 3),
                strides = 1, kernel_initializer='he_uniform',
                padding='same')(residual)
            vgg_encoded = layers.BatchNormalization(trainable=True)(vgg_encoded)
            vgg_encoded = keras.activations.relu(vgg_encoded)
            vgg_encoded = layers.Conv2D(32, (3, 3),activation="relu",␣
↪kernel_initializer='he_uniform',
                padding='same')(vgg_encoded)
            vgg_encoded = layers.BatchNormalization(trainable=True)(vgg_encoded)
            residual = layers.Add()([vgg_encoded,residual])
    # Phase 3:
    for layer in range(1,n+1):
        if layer == 1 :
            vgg_encoded = layers.Conv2D(64, (3, 3),
                strides = 2, kernel_initializer='he_uniform',
                padding='same')(residual)
            vgg_encoded = layers.BatchNormalization(trainable=True)(vgg_encoded)
            vgg_encoded = keras.activations.relu(vgg_encoded)
            vgg_encoded = layers.Conv2D(64, (3, 3),activation="relu",␣
↪kernel_initializer='he_uniform',
                padding='same')(vgg_encoded)
            vgg_encoded = layers.BatchNormalization(trainable=True)(vgg_encoded)
            residual = layers.Conv2D(64, (3, 3),
                strides = 2, kernel_initializer='he_uniform',
                padding='same')(residual)
            residual = layers.Add()([vgg_encoded,residual])
        else :
            vgg_encoded = layers.Conv2D(64, (3, 3),
                strides = 1, kernel_initializer='he_uniform',
                padding='same')(residual)
            vgg_encoded = layers.BatchNormalization(trainable=True)(vgg_encoded)
            vgg_encoded = keras.activations.relu(vgg_encoded)
            vgg_encoded = layers.Conv2D(64, (3, 3),activation="relu",␣
↪kernel_initializer='he_uniform',
                padding='same')(vgg_encoded)
            vgg_encoded = layers.BatchNormalization(trainable=True)(vgg_encoded)
            residual = layers.Add()([vgg_encoded,residual])
    residual = layers.GlobalAveragePooling2D()(residual)
    representation = layers.Flatten()(residual)
    representation = layers.Dropout(0.5)(representation)
    features = layers.
↪Dense(64,activation="relu",kernel_initializer="he_uniform")(representation)
    logits = layers.Dense(num_classes)(features)
    # Create the Keras model.
    model = keras.Model(inputs=inputs, outputs=logits)
    return model
```

Each model were trained with the AdamW optimizer, a learning rate of $1e-3$ and a decaying weight of $1e-4$. The training was run for 100 epochs with an early stopping based on the validation accuracy and a mini-batch size of 128.

[20]:
```
def run_experiment(model):
    optimizer = tfa.optimizers.AdamW(
        learning_rate=learning_rate, weight_decay=weight_decay
    )

    model.compile(
        optimizer=optimizer,
        loss=keras.losses.CategoricalCrossentropy(from_logits=True),
        metrics=[
            keras.metrics.CategoricalAccuracy(name="accuracy"),
            keras.metrics.TopKCategoricalAccuracy(5, name="top-5-accuracy"),
        ],
    )

    checkpoint_filepath = "./tmp/checkpoint_"+str(model.name)
    checkpoint_callback = keras.callbacks.ModelCheckpoint(
        checkpoint_filepath,
        monitor="val_accuracy",
        save_best_only=True,
        save_weights_only=True,
    )
    early_stoping_callback = keras.callbacks.
↪EarlyStopping(monitor='val_accuracy', patience=15)
    history = model.fit(
        x=train,
        y=train_label,
        batch_size=batch_size,
        epochs=num_epochs,
        validation_split=0.1,
        callbacks=[checkpoint_callback,early_stoping_callback],
    )

    model.load_weights(checkpoint_filepath)
    _, accuracy, top_5_accuracy = model.evaluate(test, test_labels)
    print(f"Test accuracy: {round(accuracy * 100, 2)}%")
    print(f"Test top 5 accuracy: {round(top_5_accuracy * 100, 2)}%")

    return history
```

1.4 Training VGG

[37]:
```
vgg_classifier = create_vgg_classifier(3)
vgg_classifier._name = 'model_vgg_classifier'
history_vgg_classifier = run_experiment(vgg_classifier)
```

```
Epoch 1/100
317/317 [==============================] - 7s 17ms/step - loss: 2.2947 -
accuracy: 0.1386 - top-5-accuracy: 0.5646 - val_loss: 2.2673 - val_accuracy:
0.1709 - val_top-5-accuracy: 0.6153
Epoch 2/100
317/317 [==============================] - 5s 15ms/step - loss: 2.2691 -
accuracy: 0.1684 - top-5-accuracy: 0.6045 - val_loss: 2.2564 - val_accuracy:
0.1787 - val_top-5-accuracy: 0.6058
Epoch 3/100
317/317 [==============================] - 5s 15ms/step - loss: 2.2569 -
accuracy: 0.1849 - top-5-accuracy: 0.6157 - val_loss: 2.2451 - val_accuracy:
0.2036 - val_top-5-accuracy: 0.6311
Epoch 4/100
317/317 [==============================] - 5s 15ms/step - loss: 2.2500 -
accuracy: 0.1926 - top-5-accuracy: 0.6238 - val_loss: 2.2399 - val_accuracy:
0.1973 - val_top-5-accuracy: 0.6200
Epoch 5/100
317/317 [==============================] - 5s 15ms/step - loss: 2.2416 -
accuracy: 0.2025 - top-5-accuracy: 0.6258 - val_loss: 2.2256 - val_accuracy:
0.2124 - val_top-5-accuracy: 0.6358
Epoch 6/100
```

```
317/317 [==============================] - 5s 15ms/step - loss: 2.2323 -
accuracy: 0.2114 - top-5-accuracy: 0.6334 - val_loss: 2.2178 - val_accuracy:
0.2176 - val_top-5-accuracy: 0.6429
Epoch 7/100
317/317 [==============================] - 5s 15ms/step - loss: 2.2277 -
accuracy: 0.2180 - top-5-accuracy: 0.6352 - val_loss: 2.2100 - val_accuracy:
0.2264 - val_top-5-accuracy: 0.6442
Epoch 8/100
317/317 [==============================] - 5s 15ms/step - loss: 2.2214 -
accuracy: 0.2234 - top-5-accuracy: 0.6402 - val_loss: 2.2119 - val_accuracy:
0.2364 - val_top-5-accuracy: 0.6398
Epoch 9/100
317/317 [==============================] - 5s 15ms/step - loss: 2.2170 -
accuracy: 0.2285 - top-5-accuracy: 0.6395 - val_loss: 2.1995 - val_accuracy:
0.2482 - val_top-5-accuracy: 0.6440
Epoch 10/100
317/317 [==============================] - 5s 15ms/step - loss: 2.2128 -
accuracy: 0.2334 - top-5-accuracy: 0.6434 - val_loss: 2.2026 - val_accuracy:
0.2464 - val_top-5-accuracy: 0.6393
Epoch 11/100
317/317 [==============================] - 5s 16ms/step - loss: 2.2071 -
accuracy: 0.2418 - top-5-accuracy: 0.6471 - val_loss: 2.1998 - val_accuracy:
0.2538 - val_top-5-accuracy: 0.6433
Epoch 12/100
317/317 [==============================] - 5s 15ms/step - loss: 2.2021 -
accuracy: 0.2467 - top-5-accuracy: 0.6513 - val_loss: 2.2029 - val_accuracy:
0.2520 - val_top-5-accuracy: 0.6493
Epoch 13/100
317/317 [==============================] - 5s 15ms/step - loss: 2.2001 -
accuracy: 0.2490 - top-5-accuracy: 0.6498 - val_loss: 2.2101 - val_accuracy:
0.2422 - val_top-5-accuracy: 0.6358
Epoch 14/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1946 -
accuracy: 0.2507 - top-5-accuracy: 0.6545 - val_loss: 2.2008 - val_accuracy:
0.2553 - val_top-5-accuracy: 0.6409
Epoch 15/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1915 -
accuracy: 0.2550 - top-5-accuracy: 0.6562 - val_loss: 2.1947 - val_accuracy:
0.2642 - val_top-5-accuracy: 0.6402
Epoch 16/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1907 -
accuracy: 0.2531 - top-5-accuracy: 0.6572 - val_loss: 2.1952 - val_accuracy:
0.2547 - val_top-5-accuracy: 0.6522
Epoch 17/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1848 -
accuracy: 0.2598 - top-5-accuracy: 0.6611 - val_loss: 2.2056 - val_accuracy:
0.2498 - val_top-5-accuracy: 0.6358
Epoch 18/100
```

```
317/317 [==============================] - 5s 15ms/step - loss: 2.1797 -
accuracy: 0.2640 - top-5-accuracy: 0.6642 - val_loss: 2.1989 - val_accuracy:
0.2611 - val_top-5-accuracy: 0.6482
Epoch 19/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1794 -
accuracy: 0.2634 - top-5-accuracy: 0.6663 - val_loss: 2.1871 - val_accuracy:
0.2702 - val_top-5-accuracy: 0.6458
Epoch 20/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1754 -
accuracy: 0.2693 - top-5-accuracy: 0.6669 - val_loss: 2.1892 - val_accuracy:
0.2771 - val_top-5-accuracy: 0.6451
Epoch 21/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1722 -
accuracy: 0.2695 - top-5-accuracy: 0.6725 - val_loss: 2.2022 - val_accuracy:
0.2609 - val_top-5-accuracy: 0.6316
Epoch 22/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1702 -
accuracy: 0.2724 - top-5-accuracy: 0.6741 - val_loss: 2.1868 - val_accuracy:
0.2696 - val_top-5-accuracy: 0.6462
Epoch 23/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1644 -
accuracy: 0.2750 - top-5-accuracy: 0.6758 - val_loss: 2.2040 - val_accuracy:
0.2629 - val_top-5-accuracy: 0.6427
Epoch 24/100
317/317 [==============================] - 5s 16ms/step - loss: 2.1641 -
accuracy: 0.2736 - top-5-accuracy: 0.6778 - val_loss: 2.1816 - val_accuracy:
0.2798 - val_top-5-accuracy: 0.6556
Epoch 25/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1590 -
accuracy: 0.2793 - top-5-accuracy: 0.6823 - val_loss: 2.1822 - val_accuracy:
0.2798 - val_top-5-accuracy: 0.6367
Epoch 26/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1576 -
accuracy: 0.2804 - top-5-accuracy: 0.6809 - val_loss: 2.1831 - val_accuracy:
0.2787 - val_top-5-accuracy: 0.6402
Epoch 27/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1558 -
accuracy: 0.2799 - top-5-accuracy: 0.6815 - val_loss: 2.1915 - val_accuracy:
0.2729 - val_top-5-accuracy: 0.6424
Epoch 28/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1525 -
accuracy: 0.2835 - top-5-accuracy: 0.6864 - val_loss: 2.1865 - val_accuracy:
0.2802 - val_top-5-accuracy: 0.6389
Epoch 29/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1481 -
accuracy: 0.2848 - top-5-accuracy: 0.6905 - val_loss: 2.1962 - val_accuracy:
0.2731 - val_top-5-accuracy: 0.6447
Epoch 30/100
```

```
317/317 [==============================] - 5s 15ms/step - loss: 2.1484 -
accuracy: 0.2861 - top-5-accuracy: 0.6924 - val_loss: 2.2001 - val_accuracy:
0.2742 - val_top-5-accuracy: 0.6478
Epoch 31/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1450 -
accuracy: 0.2868 - top-5-accuracy: 0.6900 - val_loss: 2.1941 - val_accuracy:
0.2767 - val_top-5-accuracy: 0.6504
Epoch 32/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1393 -
accuracy: 0.2916 - top-5-accuracy: 0.6961 - val_loss: 2.1939 - val_accuracy:
0.2791 - val_top-5-accuracy: 0.6429
Epoch 33/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1395 -
accuracy: 0.2913 - top-5-accuracy: 0.6966 - val_loss: 2.1824 - val_accuracy:
0.2849 - val_top-5-accuracy: 0.6498
Epoch 34/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1383 -
accuracy: 0.2879 - top-5-accuracy: 0.6975 - val_loss: 2.1862 - val_accuracy:
0.2816 - val_top-5-accuracy: 0.6484
Epoch 35/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1356 -
accuracy: 0.2918 - top-5-accuracy: 0.6978 - val_loss: 2.1910 - val_accuracy:
0.2798 - val_top-5-accuracy: 0.6404
Epoch 36/100
317/317 [==============================] - 5s 16ms/step - loss: 2.1315 -
accuracy: 0.2920 - top-5-accuracy: 0.7027 - val_loss: 2.1857 - val_accuracy:
0.2829 - val_top-5-accuracy: 0.6482
Epoch 37/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1296 -
accuracy: 0.2940 - top-5-accuracy: 0.7011 - val_loss: 2.2153 - val_accuracy:
0.2676 - val_top-5-accuracy: 0.6351
Epoch 38/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1303 -
accuracy: 0.2939 - top-5-accuracy: 0.7006 - val_loss: 2.1921 - val_accuracy:
0.2791 - val_top-5-accuracy: 0.6453
Epoch 39/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1262 -
accuracy: 0.2958 - top-5-accuracy: 0.7047 - val_loss: 2.1998 - val_accuracy:
0.2798 - val_top-5-accuracy: 0.6382
Epoch 40/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1221 -
accuracy: 0.2974 - top-5-accuracy: 0.7070 - val_loss: 2.1905 - val_accuracy:
0.2787 - val_top-5-accuracy: 0.6487
Epoch 41/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1206 -
accuracy: 0.2979 - top-5-accuracy: 0.7129 - val_loss: 2.1850 - val_accuracy:
0.2818 - val_top-5-accuracy: 0.6447
Epoch 42/100
```

```
317/317 [==============================] - 5s 15ms/step - loss: 2.1205 -
accuracy: 0.2981 - top-5-accuracy: 0.7124 - val_loss: 2.2010 - val_accuracy:
0.2776 - val_top-5-accuracy: 0.6380
Epoch 43/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1196 -
accuracy: 0.2985 - top-5-accuracy: 0.7089 - val_loss: 2.1984 - val_accuracy:
0.2747 - val_top-5-accuracy: 0.6353
Epoch 44/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1134 -
accuracy: 0.3011 - top-5-accuracy: 0.7153 - val_loss: 2.2159 - val_accuracy:
0.2620 - val_top-5-accuracy: 0.6300
Epoch 45/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1127 -
accuracy: 0.3012 - top-5-accuracy: 0.7123 - val_loss: 2.2014 - val_accuracy:
0.2767 - val_top-5-accuracy: 0.6513
Epoch 46/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1121 -
accuracy: 0.3011 - top-5-accuracy: 0.7162 - val_loss: 2.1948 - val_accuracy:
0.2811 - val_top-5-accuracy: 0.6436
Epoch 47/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1098 -
accuracy: 0.3026 - top-5-accuracy: 0.7182 - val_loss: 2.1944 - val_accuracy:
0.2778 - val_top-5-accuracy: 0.6449
Epoch 48/100
317/317 [==============================] - 5s 15ms/step - loss: 2.1045 -
accuracy: 0.3054 - top-5-accuracy: 0.7200 - val_loss: 2.2107 - val_accuracy:
0.2733 - val_top-5-accuracy: 0.6438
157/157 [==============================] - 1s 4ms/step - loss: 1.6631 -
accuracy: 0.6604 - top-5-accuracy: 0.9406
Test accuracy: 66.04%
Test top 5 accuracy: 94.06%
```

1.5 Training CCT

[39]:
```
cct_classifier = create_cct_classifier()
cct_classifier._name = 'model_cct_classifier'
history_cct_classifier = run_experiment(cct_classifier)
```

```
Epoch 1/100
317/317 [==============================] - 13s 31ms/step - loss: 2.3268 -
accuracy: 0.1188 - top-5-accuracy: 0.5371 - val_loss: 2.3013 - val_accuracy:
0.1164 - val_top-5-accuracy: 0.5507
Epoch 2/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2874 -
accuracy: 0.1374 - top-5-accuracy: 0.5821 - val_loss: 2.2751 - val_accuracy:
0.1420 - val_top-5-accuracy: 0.6047
Epoch 3/100
317/317 [==============================] - 9s 29ms/step - loss: 2.2763 -
```

```
accuracy: 0.1502 - top-5-accuracy: 0.5999 - val_loss: 2.2632 - val_accuracy:
0.1633 - val_top-5-accuracy: 0.6151
Epoch 4/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2687 -
accuracy: 0.1633 - top-5-accuracy: 0.6064 - val_loss: 2.2493 - val_accuracy:
0.1798 - val_top-5-accuracy: 0.6364
Epoch 5/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2617 -
accuracy: 0.1712 - top-5-accuracy: 0.6140 - val_loss: 2.2433 - val_accuracy:
0.1944 - val_top-5-accuracy: 0.6367
Epoch 6/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2532 -
accuracy: 0.1849 - top-5-accuracy: 0.6212 - val_loss: 2.2379 - val_accuracy:
0.1984 - val_top-5-accuracy: 0.6436
Epoch 7/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2472 -
accuracy: 0.1933 - top-5-accuracy: 0.6229 - val_loss: 2.2373 - val_accuracy:
0.2053 - val_top-5-accuracy: 0.6316
Epoch 8/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2409 -
accuracy: 0.2002 - top-5-accuracy: 0.6305 - val_loss: 2.2278 - val_accuracy:
0.2142 - val_top-5-accuracy: 0.6398
Epoch 9/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2357 -
accuracy: 0.2104 - top-5-accuracy: 0.6321 - val_loss: 2.2222 - val_accuracy:
0.2262 - val_top-5-accuracy: 0.6358
Epoch 10/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2307 -
accuracy: 0.2143 - top-5-accuracy: 0.6338 - val_loss: 2.2236 - val_accuracy:
0.2216 - val_top-5-accuracy: 0.6438
Epoch 11/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2254 -
accuracy: 0.2181 - top-5-accuracy: 0.6361 - val_loss: 2.2212 - val_accuracy:
0.2287 - val_top-5-accuracy: 0.6420
Epoch 12/100
317/317 [==============================] - 9s 27ms/step - loss: 2.2217 -
accuracy: 0.2222 - top-5-accuracy: 0.6396 - val_loss: 2.2298 - val_accuracy:
0.2133 - val_top-5-accuracy: 0.6324
Epoch 13/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2181 -
accuracy: 0.2266 - top-5-accuracy: 0.6392 - val_loss: 2.2149 - val_accuracy:
0.2362 - val_top-5-accuracy: 0.6427
Epoch 14/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2138 -
accuracy: 0.2328 - top-5-accuracy: 0.6425 - val_loss: 2.2237 - val_accuracy:
0.2329 - val_top-5-accuracy: 0.6404
Epoch 15/100
317/317 [==============================] - 9s 27ms/step - loss: 2.2095 -
```

```
accuracy: 0.2372 - top-5-accuracy: 0.6443 - val_loss: 2.2071 - val_accuracy:
0.2358 - val_top-5-accuracy: 0.6451
Epoch 16/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2060 -
accuracy: 0.2398 - top-5-accuracy: 0.6472 - val_loss: 2.2117 - val_accuracy:
0.2380 - val_top-5-accuracy: 0.6436
Epoch 17/100
317/317 [==============================] - 9s 28ms/step - loss: 2.2003 -
accuracy: 0.2443 - top-5-accuracy: 0.6521 - val_loss: 2.2118 - val_accuracy:
0.2378 - val_top-5-accuracy: 0.6387
Epoch 18/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1978 -
accuracy: 0.2463 - top-5-accuracy: 0.6547 - val_loss: 2.2192 - val_accuracy:
0.2356 - val_top-5-accuracy: 0.6311
Epoch 19/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1970 -
accuracy: 0.2472 - top-5-accuracy: 0.6526 - val_loss: 2.2090 - val_accuracy:
0.2378 - val_top-5-accuracy: 0.6389
Epoch 20/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1924 -
accuracy: 0.2511 - top-5-accuracy: 0.6547 - val_loss: 2.2036 - val_accuracy:
0.2518 - val_top-5-accuracy: 0.6464
Epoch 21/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1894 -
accuracy: 0.2535 - top-5-accuracy: 0.6586 - val_loss: 2.2126 - val_accuracy:
0.2429 - val_top-5-accuracy: 0.6364
Epoch 22/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1872 -
accuracy: 0.2572 - top-5-accuracy: 0.6579 - val_loss: 2.1943 - val_accuracy:
0.2578 - val_top-5-accuracy: 0.6433
Epoch 23/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1837 -
accuracy: 0.2581 - top-5-accuracy: 0.6605 - val_loss: 2.1940 - val_accuracy:
0.2598 - val_top-5-accuracy: 0.6491
Epoch 24/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1815 -
accuracy: 0.2597 - top-5-accuracy: 0.6620 - val_loss: 2.1881 - val_accuracy:
0.2664 - val_top-5-accuracy: 0.6487
Epoch 25/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1785 -
accuracy: 0.2641 - top-5-accuracy: 0.6633 - val_loss: 2.1955 - val_accuracy:
0.2569 - val_top-5-accuracy: 0.6458
Epoch 26/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1738 -
accuracy: 0.2691 - top-5-accuracy: 0.6658 - val_loss: 2.1879 - val_accuracy:
0.2664 - val_top-5-accuracy: 0.6451
Epoch 27/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1718 -
```

```
accuracy: 0.2689 - top-5-accuracy: 0.6681 - val_loss: 2.1885 - val_accuracy:
0.2684 - val_top-5-accuracy: 0.6491
Epoch 28/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1692 -
accuracy: 0.2724 - top-5-accuracy: 0.6681 - val_loss: 2.1993 - val_accuracy:
0.2551 - val_top-5-accuracy: 0.6378
Epoch 29/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1662 -
accuracy: 0.2751 - top-5-accuracy: 0.6705 - val_loss: 2.1802 - val_accuracy:
0.2727 - val_top-5-accuracy: 0.6491
Epoch 30/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1629 -
accuracy: 0.2760 - top-5-accuracy: 0.6705 - val_loss: 2.1951 - val_accuracy:
0.2649 - val_top-5-accuracy: 0.6376
Epoch 31/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1616 -
accuracy: 0.2782 - top-5-accuracy: 0.6733 - val_loss: 2.1973 - val_accuracy:
0.2607 - val_top-5-accuracy: 0.6449
Epoch 32/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1596 -
accuracy: 0.2785 - top-5-accuracy: 0.6747 - val_loss: 2.2120 - val_accuracy:
0.2420 - val_top-5-accuracy: 0.6356
Epoch 33/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1573 -
accuracy: 0.2815 - top-5-accuracy: 0.6777 - val_loss: 2.1959 - val_accuracy:
0.2622 - val_top-5-accuracy: 0.6413
Epoch 34/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1554 -
accuracy: 0.2823 - top-5-accuracy: 0.6784 - val_loss: 2.1932 - val_accuracy:
0.2647 - val_top-5-accuracy: 0.6396
Epoch 35/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1524 -
accuracy: 0.2837 - top-5-accuracy: 0.6797 - val_loss: 2.1874 - val_accuracy:
0.2711 - val_top-5-accuracy: 0.6482
Epoch 36/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1517 -
accuracy: 0.2852 - top-5-accuracy: 0.6817 - val_loss: 2.1951 - val_accuracy:
0.2593 - val_top-5-accuracy: 0.6280
Epoch 37/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1479 -
accuracy: 0.2864 - top-5-accuracy: 0.6839 - val_loss: 2.1857 - val_accuracy:
0.2702 - val_top-5-accuracy: 0.6482
Epoch 38/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1460 -
accuracy: 0.2887 - top-5-accuracy: 0.6856 - val_loss: 2.1938 - val_accuracy:
0.2664 - val_top-5-accuracy: 0.6411
Epoch 39/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1424 -
```

```
accuracy: 0.2905 - top-5-accuracy: 0.6883 - val_loss: 2.1848 - val_accuracy:
0.2753 - val_top-5-accuracy: 0.6387
Epoch 40/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1413 -
accuracy: 0.2907 - top-5-accuracy: 0.6874 - val_loss: 2.2020 - val_accuracy:
0.2627 - val_top-5-accuracy: 0.6362
Epoch 41/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1379 -
accuracy: 0.2927 - top-5-accuracy: 0.6883 - val_loss: 2.1886 - val_accuracy:
0.2669 - val_top-5-accuracy: 0.6502
Epoch 42/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1379 -
accuracy: 0.2932 - top-5-accuracy: 0.6901 - val_loss: 2.1945 - val_accuracy:
0.2689 - val_top-5-accuracy: 0.6420
Epoch 43/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1346 -
accuracy: 0.2953 - top-5-accuracy: 0.6926 - val_loss: 2.1968 - val_accuracy:
0.2613 - val_top-5-accuracy: 0.6349
Epoch 44/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1332 -
accuracy: 0.2970 - top-5-accuracy: 0.6915 - val_loss: 2.1845 - val_accuracy:
0.2747 - val_top-5-accuracy: 0.6498
Epoch 45/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1314 -
accuracy: 0.2968 - top-5-accuracy: 0.6958 - val_loss: 2.1994 - val_accuracy:
0.2636 - val_top-5-accuracy: 0.6391
Epoch 46/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1300 -
accuracy: 0.2995 - top-5-accuracy: 0.6951 - val_loss: 2.2050 - val_accuracy:
0.2676 - val_top-5-accuracy: 0.6353
Epoch 47/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1251 -
accuracy: 0.3003 - top-5-accuracy: 0.6994 - val_loss: 2.1971 - val_accuracy:
0.2667 - val_top-5-accuracy: 0.6322
Epoch 48/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1248 -
accuracy: 0.3006 - top-5-accuracy: 0.7010 - val_loss: 2.2061 - val_accuracy:
0.2664 - val_top-5-accuracy: 0.6380
Epoch 49/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1243 -
accuracy: 0.3014 - top-5-accuracy: 0.6963 - val_loss: 2.2027 - val_accuracy:
0.2671 - val_top-5-accuracy: 0.6391
Epoch 50/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1224 -
accuracy: 0.3009 - top-5-accuracy: 0.6986 - val_loss: 2.2099 - val_accuracy:
0.2684 - val_top-5-accuracy: 0.6347
Epoch 51/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1185 -
```

```
accuracy: 0.3039 - top-5-accuracy: 0.7022 - val_loss: 2.2070 - val_accuracy:
0.2700 - val_top-5-accuracy: 0.6353
Epoch 52/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1179 -
accuracy: 0.3050 - top-5-accuracy: 0.7037 - val_loss: 2.2003 - val_accuracy:
0.2747 - val_top-5-accuracy: 0.6336
Epoch 53/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1146 -
accuracy: 0.3057 - top-5-accuracy: 0.7064 - val_loss: 2.1947 - val_accuracy:
0.2762 - val_top-5-accuracy: 0.6371
Epoch 54/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1123 -
accuracy: 0.3075 - top-5-accuracy: 0.7061 - val_loss: 2.1887 - val_accuracy:
0.2807 - val_top-5-accuracy: 0.6442
Epoch 55/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1097 -
accuracy: 0.3085 - top-5-accuracy: 0.7082 - val_loss: 2.2028 - val_accuracy:
0.2711 - val_top-5-accuracy: 0.6387
Epoch 56/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1093 -
accuracy: 0.3106 - top-5-accuracy: 0.7066 - val_loss: 2.2027 - val_accuracy:
0.2753 - val_top-5-accuracy: 0.6322
Epoch 57/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1101 -
accuracy: 0.3079 - top-5-accuracy: 0.7064 - val_loss: 2.1948 - val_accuracy:
0.2753 - val_top-5-accuracy: 0.6513
Epoch 58/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1054 -
accuracy: 0.3107 - top-5-accuracy: 0.7097 - val_loss: 2.2051 - val_accuracy:
0.2658 - val_top-5-accuracy: 0.6369
Epoch 59/100
317/317 [==============================] - 9s 28ms/step - loss: 2.1043 -
accuracy: 0.3115 - top-5-accuracy: 0.7108 - val_loss: 2.2056 - val_accuracy:
0.2636 - val_top-5-accuracy: 0.6344
Epoch 60/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1023 -
accuracy: 0.3127 - top-5-accuracy: 0.7125 - val_loss: 2.2099 - val_accuracy:
0.2689 - val_top-5-accuracy: 0.6302
Epoch 61/100
317/317 [==============================] - 9s 27ms/step - loss: 2.1022 -
accuracy: 0.3111 - top-5-accuracy: 0.7132 - val_loss: 2.2042 - val_accuracy:
0.2738 - val_top-5-accuracy: 0.6293
Epoch 62/100
317/317 [==============================] - 9s 27ms/step - loss: 2.0969 -
accuracy: 0.3151 - top-5-accuracy: 0.7161 - val_loss: 2.2130 - val_accuracy:
0.2624 - val_top-5-accuracy: 0.6209
Epoch 63/100
317/317 [==============================] - 9s 27ms/step - loss: 2.0970 -
```

```
accuracy: 0.3157 - top-5-accuracy: 0.7148 - val_loss: 2.2021 - val_accuracy:
0.2722 - val_top-5-accuracy: 0.6369
Epoch 64/100
317/317 [==============================] - 9s 27ms/step - loss: 2.0972 -
accuracy: 0.3146 - top-5-accuracy: 0.7153 - val_loss: 2.2119 - val_accuracy:
0.2731 - val_top-5-accuracy: 0.6387
Epoch 65/100
317/317 [==============================] - 9s 28ms/step - loss: 2.0936 -
accuracy: 0.3166 - top-5-accuracy: 0.7185 - val_loss: 2.2150 - val_accuracy:
0.2662 - val_top-5-accuracy: 0.6360
Epoch 66/100
317/317 [==============================] - 9s 28ms/step - loss: 2.0942 -
accuracy: 0.3176 - top-5-accuracy: 0.7159 - val_loss: 2.2054 - val_accuracy:
0.2733 - val_top-5-accuracy: 0.6431
Epoch 67/100
317/317 [==============================] - 9s 27ms/step - loss: 2.0910 -
accuracy: 0.3165 - top-5-accuracy: 0.7196 - val_loss: 2.2039 - val_accuracy:
0.2671 - val_top-5-accuracy: 0.6396
Epoch 68/100
317/317 [==============================] - 9s 27ms/step - loss: 2.0885 -
accuracy: 0.3205 - top-5-accuracy: 0.7204 - val_loss: 2.2056 - val_accuracy:
0.2698 - val_top-5-accuracy: 0.6422
Epoch 69/100
317/317 [==============================] - 9s 27ms/step - loss: 2.0879 -
accuracy: 0.3188 - top-5-accuracy: 0.7218 - val_loss: 2.2146 - val_accuracy:
0.2698 - val_top-5-accuracy: 0.6340
157/157 [==============================] - 1s 7ms/step - loss: 1.6608 -
accuracy: 0.6202 - top-5-accuracy: 0.9296
Test accuracy: 62.02%
Test top 5 accuracy: 92.96%
```

1.6 Training ResNet20

```
[41]: resnet20_classifier = create_resnet_classifier(1)
      resnet20_classifier._name = 'model_resnet20_classifier'
      history_resnet20_classifier = run_experiment(resnet20_classifier)
```

```
Epoch 1/100
317/317 [==============================] - 8s 19ms/step - loss: 2.3501 -
accuracy: 0.1247 - top-5-accuracy: 0.5516 - val_loss: 2.2990 - val_accuracy:
0.1244 - val_top-5-accuracy: 0.5478
Epoch 2/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2879 -
accuracy: 0.1457 - top-5-accuracy: 0.5851 - val_loss: 2.2680 - val_accuracy:
0.1727 - val_top-5-accuracy: 0.6171
Epoch 3/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2731 -
accuracy: 0.1608 - top-5-accuracy: 0.6038 - val_loss: 2.2590 - val_accuracy:
```

```
0.1698 - val_top-5-accuracy: 0.6287
Epoch 4/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2650 -
accuracy: 0.1722 - top-5-accuracy: 0.6108 - val_loss: 2.2451 - val_accuracy:
0.1909 - val_top-5-accuracy: 0.6309
Epoch 5/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2600 -
accuracy: 0.1810 - top-5-accuracy: 0.6139 - val_loss: 2.2384 - val_accuracy:
0.2071 - val_top-5-accuracy: 0.6384
Epoch 6/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2533 -
accuracy: 0.1905 - top-5-accuracy: 0.6221 - val_loss: 2.2365 - val_accuracy:
0.2122 - val_top-5-accuracy: 0.6336
Epoch 7/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2466 -
accuracy: 0.1965 - top-5-accuracy: 0.6265 - val_loss: 2.2614 - val_accuracy:
0.1956 - val_top-5-accuracy: 0.6300
Epoch 8/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2413 -
accuracy: 0.2049 - top-5-accuracy: 0.6259 - val_loss: 2.2548 - val_accuracy:
0.1838 - val_top-5-accuracy: 0.6207
Epoch 9/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2388 -
accuracy: 0.2094 - top-5-accuracy: 0.6283 - val_loss: 2.3087 - val_accuracy:
0.1827 - val_top-5-accuracy: 0.6164
Epoch 10/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2346 -
accuracy: 0.2135 - top-5-accuracy: 0.6342 - val_loss: 2.2221 - val_accuracy:
0.2267 - val_top-5-accuracy: 0.6369
Epoch 11/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2310 -
accuracy: 0.2169 - top-5-accuracy: 0.6333 - val_loss: 2.2464 - val_accuracy:
0.2036 - val_top-5-accuracy: 0.6253
Epoch 12/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2282 -
accuracy: 0.2219 - top-5-accuracy: 0.6347 - val_loss: 2.2442 - val_accuracy:
0.2020 - val_top-5-accuracy: 0.6416
Epoch 13/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2235 -
accuracy: 0.2270 - top-5-accuracy: 0.6381 - val_loss: 2.2273 - val_accuracy:
0.2162 - val_top-5-accuracy: 0.6324
Epoch 14/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2218 -
accuracy: 0.2300 - top-5-accuracy: 0.6404 - val_loss: 2.2427 - val_accuracy:
0.2107 - val_top-5-accuracy: 0.6187
Epoch 15/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2169 -
accuracy: 0.2335 - top-5-accuracy: 0.6427 - val_loss: 2.2126 - val_accuracy:
```

```
0.2427 - val_top-5-accuracy: 0.6362
Epoch 16/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2158 -
accuracy: 0.2365 - top-5-accuracy: 0.6402 - val_loss: 2.2361 - val_accuracy:
0.2191 - val_top-5-accuracy: 0.6249
Epoch 17/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2097 -
accuracy: 0.2404 - top-5-accuracy: 0.6453 - val_loss: 2.2445 - val_accuracy:
0.2118 - val_top-5-accuracy: 0.6373
Epoch 18/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2093 -
accuracy: 0.2406 - top-5-accuracy: 0.6451 - val_loss: 2.2557 - val_accuracy:
0.1840 - val_top-5-accuracy: 0.6218
Epoch 19/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2075 -
accuracy: 0.2455 - top-5-accuracy: 0.6460 - val_loss: 2.2560 - val_accuracy:
0.2120 - val_top-5-accuracy: 0.6260
Epoch 20/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2049 -
accuracy: 0.2471 - top-5-accuracy: 0.6486 - val_loss: 2.2192 - val_accuracy:
0.2336 - val_top-5-accuracy: 0.6247
Epoch 21/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2024 -
accuracy: 0.2495 - top-5-accuracy: 0.6502 - val_loss: 2.1948 - val_accuracy:
0.2569 - val_top-5-accuracy: 0.6416
Epoch 22/100
317/317 [==============================] - 5s 17ms/step - loss: 2.2026 -
accuracy: 0.2520 - top-5-accuracy: 0.6472 - val_loss: 2.1996 - val_accuracy:
0.2582 - val_top-5-accuracy: 0.6413
Epoch 23/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1992 -
accuracy: 0.2520 - top-5-accuracy: 0.6487 - val_loss: 2.2103 - val_accuracy:
0.2344 - val_top-5-accuracy: 0.6413
Epoch 24/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1985 -
accuracy: 0.2543 - top-5-accuracy: 0.6471 - val_loss: 2.2098 - val_accuracy:
0.2398 - val_top-5-accuracy: 0.6409
Epoch 25/100
317/317 [==============================] - 6s 18ms/step - loss: 2.1958 -
accuracy: 0.2546 - top-5-accuracy: 0.6494 - val_loss: 2.1915 - val_accuracy:
0.2624 - val_top-5-accuracy: 0.6462
Epoch 26/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1947 -
accuracy: 0.2604 - top-5-accuracy: 0.6497 - val_loss: 2.2777 - val_accuracy:
0.2071 - val_top-5-accuracy: 0.6100
Epoch 27/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1937 -
accuracy: 0.2586 - top-5-accuracy: 0.6503 - val_loss: 2.2075 - val_accuracy:
```

```
0.2418 - val_top-5-accuracy: 0.6387
Epoch 28/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1924 -
accuracy: 0.2615 - top-5-accuracy: 0.6523 - val_loss: 2.2034 - val_accuracy:
0.2553 - val_top-5-accuracy: 0.6420
Epoch 29/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1899 -
accuracy: 0.2611 - top-5-accuracy: 0.6551 - val_loss: 2.1890 - val_accuracy:
0.2609 - val_top-5-accuracy: 0.6409
Epoch 30/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1893 -
accuracy: 0.2643 - top-5-accuracy: 0.6507 - val_loss: 2.2132 - val_accuracy:
0.2427 - val_top-5-accuracy: 0.6293
Epoch 31/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1871 -
accuracy: 0.2659 - top-5-accuracy: 0.6523 - val_loss: 2.2497 - val_accuracy:
0.2073 - val_top-5-accuracy: 0.5922
Epoch 32/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1848 -
accuracy: 0.2674 - top-5-accuracy: 0.6549 - val_loss: 2.1973 - val_accuracy:
0.2560 - val_top-5-accuracy: 0.6440
Epoch 33/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1852 -
accuracy: 0.2670 - top-5-accuracy: 0.6514 - val_loss: 2.2175 - val_accuracy:
0.2373 - val_top-5-accuracy: 0.6333
Epoch 34/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1834 -
accuracy: 0.2697 - top-5-accuracy: 0.6539 - val_loss: 2.2234 - val_accuracy:
0.2493 - val_top-5-accuracy: 0.6236
Epoch 35/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1822 -
accuracy: 0.2705 - top-5-accuracy: 0.6545 - val_loss: 2.2016 - val_accuracy:
0.2469 - val_top-5-accuracy: 0.6373
Epoch 36/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1831 -
accuracy: 0.2710 - top-5-accuracy: 0.6545 - val_loss: 2.1914 - val_accuracy:
0.2569 - val_top-5-accuracy: 0.6431
Epoch 37/100
317/317 [==============================] - 6s 18ms/step - loss: 2.1809 -
accuracy: 0.2728 - top-5-accuracy: 0.6546 - val_loss: 2.2196 - val_accuracy:
0.2427 - val_top-5-accuracy: 0.6427
Epoch 38/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1801 -
accuracy: 0.2716 - top-5-accuracy: 0.6580 - val_loss: 2.2612 - val_accuracy:
0.2051 - val_top-5-accuracy: 0.6082
Epoch 39/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1803 -
accuracy: 0.2705 - top-5-accuracy: 0.6550 - val_loss: 2.2300 - val_accuracy:
```

```
0.2358 - val_top-5-accuracy: 0.6149
Epoch 40/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1783 -
accuracy: 0.2740 - top-5-accuracy: 0.6574 - val_loss: 2.1750 - val_accuracy:
0.2769 - val_top-5-accuracy: 0.6496
Epoch 41/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1788 -
accuracy: 0.2747 - top-5-accuracy: 0.6554 - val_loss: 2.2552 - val_accuracy:
0.2044 - val_top-5-accuracy: 0.6098
Epoch 42/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1759 -
accuracy: 0.2744 - top-5-accuracy: 0.6578 - val_loss: 2.2731 - val_accuracy:
0.1976 - val_top-5-accuracy: 0.6044
Epoch 43/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1750 -
accuracy: 0.2778 - top-5-accuracy: 0.6580 - val_loss: 2.2064 - val_accuracy:
0.2587 - val_top-5-accuracy: 0.6291
Epoch 44/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1753 -
accuracy: 0.2762 - top-5-accuracy: 0.6576 - val_loss: 2.2008 - val_accuracy:
0.2549 - val_top-5-accuracy: 0.6451
Epoch 45/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1722 -
accuracy: 0.2793 - top-5-accuracy: 0.6602 - val_loss: 2.1952 - val_accuracy:
0.2629 - val_top-5-accuracy: 0.6353
Epoch 46/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1729 -
accuracy: 0.2793 - top-5-accuracy: 0.6605 - val_loss: 2.2038 - val_accuracy:
0.2444 - val_top-5-accuracy: 0.6340
Epoch 47/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1719 -
accuracy: 0.2798 - top-5-accuracy: 0.6571 - val_loss: 2.2438 - val_accuracy:
0.2122 - val_top-5-accuracy: 0.6302
Epoch 48/100
317/317 [==============================] - 6s 17ms/step - loss: 2.1721 -
accuracy: 0.2780 - top-5-accuracy: 0.6582 - val_loss: 2.2276 - val_accuracy:
0.2298 - val_top-5-accuracy: 0.6189
Epoch 49/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1703 -
accuracy: 0.2808 - top-5-accuracy: 0.6604 - val_loss: 2.2006 - val_accuracy:
0.2556 - val_top-5-accuracy: 0.6273
Epoch 50/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1727 -
accuracy: 0.2803 - top-5-accuracy: 0.6590 - val_loss: 2.2149 - val_accuracy:
0.2429 - val_top-5-accuracy: 0.6296
Epoch 51/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1693 -
accuracy: 0.2824 - top-5-accuracy: 0.6624 - val_loss: 2.2495 - val_accuracy:
```

```
0.2213 - val_top-5-accuracy: 0.6089
Epoch 52/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1681 -
accuracy: 0.2840 - top-5-accuracy: 0.6629 - val_loss: 2.3064 - val_accuracy:
0.1964 - val_top-5-accuracy: 0.6104
Epoch 53/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1698 -
accuracy: 0.2797 - top-5-accuracy: 0.6601 - val_loss: 2.1846 - val_accuracy:
0.2680 - val_top-5-accuracy: 0.6396
Epoch 54/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1674 -
accuracy: 0.2834 - top-5-accuracy: 0.6604 - val_loss: 2.1951 - val_accuracy:
0.2544 - val_top-5-accuracy: 0.6333
Epoch 55/100
317/317 [==============================] - 5s 17ms/step - loss: 2.1671 -
accuracy: 0.2849 - top-5-accuracy: 0.6611 - val_loss: 2.1975 - val_accuracy:
0.2609 - val_top-5-accuracy: 0.6371
157/157 [==============================] - 1s 4ms/step - loss: 1.7215 -
accuracy: 0.6370 - top-5-accuracy: 0.9518
Test accuracy: 63.7%
Test top 5 accuracy: 95.18%
```

1.7 Training ResNet44

[43]:
```
resnet44_classifier = create_resnet_classifier(7)
resnet44_classifier._name = 'model_resnet44_classifier'
history_resnet44_classifier = run_experiment(resnet44_classifier)
```

```
Epoch 1/100
317/317 [==============================] - 28s 71ms/step - loss: 2.4494 -
accuracy: 0.1117 - top-5-accuracy: 0.5186 - val_loss: 2.3001 - val_accuracy:
0.1147 - val_top-5-accuracy: 0.5322
Epoch 2/100
317/317 [==============================] - 22s 68ms/step - loss: 2.3061 -
accuracy: 0.1232 - top-5-accuracy: 0.5452 - val_loss: 2.2755 - val_accuracy:
0.1489 - val_top-5-accuracy: 0.6036
Epoch 3/100
317/317 [==============================] - 21s 67ms/step - loss: 2.2909 -
accuracy: 0.1400 - top-5-accuracy: 0.5687 - val_loss: 2.2691 - val_accuracy:
0.1487 - val_top-5-accuracy: 0.6096
Epoch 4/100
317/317 [==============================] - 21s 68ms/step - loss: 2.2762 -
accuracy: 0.1549 - top-5-accuracy: 0.5973 - val_loss: 2.2579 - val_accuracy:
0.1693 - val_top-5-accuracy: 0.6164
Epoch 5/100
317/317 [==============================] - 22s 68ms/step - loss: 2.2651 -
accuracy: 0.1677 - top-5-accuracy: 0.6100 - val_loss: 2.2479 - val_accuracy:
0.1942 - val_top-5-accuracy: 0.6269
```

```
Epoch 6/100
317/317 [==============================] - 22s 68ms/step - loss: 2.2588 -
accuracy: 0.1756 - top-5-accuracy: 0.6151 - val_loss: 2.2479 - val_accuracy:
0.1991 - val_top-5-accuracy: 0.6238
Epoch 7/100
317/317 [==============================] - 21s 67ms/step - loss: 2.2531 -
accuracy: 0.1846 - top-5-accuracy: 0.6214 - val_loss: 2.2497 - val_accuracy:
0.1880 - val_top-5-accuracy: 0.6236
Epoch 8/100
317/317 [==============================] - 21s 67ms/step - loss: 2.2469 -
accuracy: 0.1977 - top-5-accuracy: 0.6249 - val_loss: 2.2361 - val_accuracy:
0.1987 - val_top-5-accuracy: 0.6364
Epoch 9/100
317/317 [==============================] - 22s 68ms/step - loss: 2.2399 -
accuracy: 0.2066 - top-5-accuracy: 0.6280 - val_loss: 2.2282 - val_accuracy:
0.2142 - val_top-5-accuracy: 0.6276
Epoch 10/100
317/317 [==============================] - 21s 67ms/step - loss: 2.2352 -
accuracy: 0.2124 - top-5-accuracy: 0.6301 - val_loss: 2.2481 - val_accuracy:
0.1996 - val_top-5-accuracy: 0.6360
Epoch 11/100
317/317 [==============================] - 21s 67ms/step - loss: 2.2295 -
accuracy: 0.2146 - top-5-accuracy: 0.6324 - val_loss: 2.2296 - val_accuracy:
0.2144 - val_top-5-accuracy: 0.6247
Epoch 12/100
317/317 [==============================] - 22s 68ms/step - loss: 2.2245 -
accuracy: 0.2244 - top-5-accuracy: 0.6352 - val_loss: 2.2133 - val_accuracy:
0.2420 - val_top-5-accuracy: 0.6504
Epoch 13/100
317/317 [==============================] - 21s 67ms/step - loss: 2.2199 -
accuracy: 0.2286 - top-5-accuracy: 0.6395 - val_loss: 2.2163 - val_accuracy:
0.2251 - val_top-5-accuracy: 0.6476
Epoch 14/100
317/317 [==============================] - 21s 67ms/step - loss: 2.2164 -
accuracy: 0.2333 - top-5-accuracy: 0.6388 - val_loss: 2.2260 - val_accuracy:
0.2182 - val_top-5-accuracy: 0.6273
Epoch 15/100
317/317 [==============================] - 21s 67ms/step - loss: 2.2133 -
accuracy: 0.2375 - top-5-accuracy: 0.6407 - val_loss: 2.2246 - val_accuracy:
0.2276 - val_top-5-accuracy: 0.6467
Epoch 16/100
317/317 [==============================] - 21s 67ms/step - loss: 2.2110 -
accuracy: 0.2396 - top-5-accuracy: 0.6430 - val_loss: 2.2276 - val_accuracy:
0.2280 - val_top-5-accuracy: 0.6371
Epoch 17/100
317/317 [==============================] - 21s 68ms/step - loss: 2.2077 -
accuracy: 0.2434 - top-5-accuracy: 0.6433 - val_loss: 2.2021 - val_accuracy:
0.2433 - val_top-5-accuracy: 0.6471
```

```
Epoch 18/100
317/317 [==============================] - 22s 68ms/step - loss: 2.2053 -
accuracy: 0.2463 - top-5-accuracy: 0.6419 - val_loss: 2.1944 - val_accuracy:
0.2598 - val_top-5-accuracy: 0.6493
Epoch 19/100
317/317 [==============================] - 21s 66ms/step - loss: 2.2007 -
accuracy: 0.2519 - top-5-accuracy: 0.6442 - val_loss: 2.2176 - val_accuracy:
0.2369 - val_top-5-accuracy: 0.6376
Epoch 20/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1973 -
accuracy: 0.2568 - top-5-accuracy: 0.6471 - val_loss: 2.2134 - val_accuracy:
0.2349 - val_top-5-accuracy: 0.6387
Epoch 21/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1915 -
accuracy: 0.2605 - top-5-accuracy: 0.6497 - val_loss: 2.2227 - val_accuracy:
0.2298 - val_top-5-accuracy: 0.6300
Epoch 22/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1915 -
accuracy: 0.2619 - top-5-accuracy: 0.6485 - val_loss: 2.2086 - val_accuracy:
0.2409 - val_top-5-accuracy: 0.6404
Epoch 23/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1870 -
accuracy: 0.2681 - top-5-accuracy: 0.6486 - val_loss: 2.2012 - val_accuracy:
0.2529 - val_top-5-accuracy: 0.6482
Epoch 24/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1836 -
accuracy: 0.2679 - top-5-accuracy: 0.6511 - val_loss: 2.2004 - val_accuracy:
0.2520 - val_top-5-accuracy: 0.6447
Epoch 25/100
317/317 [==============================] - 21s 68ms/step - loss: 2.1795 -
accuracy: 0.2724 - top-5-accuracy: 0.6532 - val_loss: 2.1906 - val_accuracy:
0.2636 - val_top-5-accuracy: 0.6569
Epoch 26/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1787 -
accuracy: 0.2763 - top-5-accuracy: 0.6536 - val_loss: 2.2132 - val_accuracy:
0.2416 - val_top-5-accuracy: 0.6247
Epoch 27/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1731 -
accuracy: 0.2814 - top-5-accuracy: 0.6563 - val_loss: 2.2015 - val_accuracy:
0.2553 - val_top-5-accuracy: 0.6331
Epoch 28/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1719 -
accuracy: 0.2823 - top-5-accuracy: 0.6552 - val_loss: 2.1978 - val_accuracy:
0.2547 - val_top-5-accuracy: 0.6356
Epoch 29/100
317/317 [==============================] - 22s 68ms/step - loss: 2.1708 -
accuracy: 0.2839 - top-5-accuracy: 0.6530 - val_loss: 2.1851 - val_accuracy:
0.2667 - val_top-5-accuracy: 0.6442
```

```
Epoch 30/100
317/317 [==============================] - 21s 68ms/step - loss: 2.1651 -
accuracy: 0.2870 - top-5-accuracy: 0.6583 - val_loss: 2.1754 - val_accuracy:
0.2784 - val_top-5-accuracy: 0.6520
Epoch 31/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1631 -
accuracy: 0.2901 - top-5-accuracy: 0.6561 - val_loss: 2.1843 - val_accuracy:
0.2620 - val_top-5-accuracy: 0.6393
Epoch 32/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1619 -
accuracy: 0.2920 - top-5-accuracy: 0.6564 - val_loss: 2.1778 - val_accuracy:
0.2782 - val_top-5-accuracy: 0.6473
Epoch 33/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1585 -
accuracy: 0.2939 - top-5-accuracy: 0.6593 - val_loss: 2.2114 - val_accuracy:
0.2431 - val_top-5-accuracy: 0.6382
Epoch 34/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1545 -
accuracy: 0.2957 - top-5-accuracy: 0.6613 - val_loss: 2.1950 - val_accuracy:
0.2544 - val_top-5-accuracy: 0.6313
Epoch 35/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1536 -
accuracy: 0.2977 - top-5-accuracy: 0.6613 - val_loss: 2.1875 - val_accuracy:
0.2676 - val_top-5-accuracy: 0.6516
Epoch 36/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1503 -
accuracy: 0.2970 - top-5-accuracy: 0.6592 - val_loss: 2.2361 - val_accuracy:
0.2173 - val_top-5-accuracy: 0.6047
Epoch 37/100
317/317 [==============================] - 21s 68ms/step - loss: 2.1473 -
accuracy: 0.3037 - top-5-accuracy: 0.6621 - val_loss: 2.1714 - val_accuracy:
0.2807 - val_top-5-accuracy: 0.6444
Epoch 38/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1462 -
accuracy: 0.3029 - top-5-accuracy: 0.6624 - val_loss: 2.1766 - val_accuracy:
0.2696 - val_top-5-accuracy: 0.6387
Epoch 39/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1448 -
accuracy: 0.3051 - top-5-accuracy: 0.6638 - val_loss: 2.1829 - val_accuracy:
0.2793 - val_top-5-accuracy: 0.6407
Epoch 40/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1409 -
accuracy: 0.3075 - top-5-accuracy: 0.6647 - val_loss: 2.1817 - val_accuracy:
0.2711 - val_top-5-accuracy: 0.6520
Epoch 41/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1388 -
accuracy: 0.3096 - top-5-accuracy: 0.6619 - val_loss: 2.1812 - val_accuracy:
0.2693 - val_top-5-accuracy: 0.6449
```

```
Epoch 42/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1361 -
accuracy: 0.3097 - top-5-accuracy: 0.6683 - val_loss: 2.1707 - val_accuracy:
0.2767 - val_top-5-accuracy: 0.6469
Epoch 43/100
317/317 [==============================] - 22s 68ms/step - loss: 2.1356 -
accuracy: 0.3113 - top-5-accuracy: 0.6671 - val_loss: 2.1703 - val_accuracy:
0.2864 - val_top-5-accuracy: 0.6464
Epoch 44/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1322 -
accuracy: 0.3147 - top-5-accuracy: 0.6684 - val_loss: 2.1747 - val_accuracy:
0.2813 - val_top-5-accuracy: 0.6429
Epoch 45/100
317/317 [==============================] - 22s 68ms/step - loss: 2.1322 -
accuracy: 0.3138 - top-5-accuracy: 0.6674 - val_loss: 2.1596 - val_accuracy:
0.2962 - val_top-5-accuracy: 0.6576
Epoch 46/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1318 -
accuracy: 0.3151 - top-5-accuracy: 0.6683 - val_loss: 2.1642 - val_accuracy:
0.2876 - val_top-5-accuracy: 0.6538
Epoch 47/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1272 -
accuracy: 0.3181 - top-5-accuracy: 0.6695 - val_loss: 2.1827 - val_accuracy:
0.2756 - val_top-5-accuracy: 0.6418
Epoch 48/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1244 -
accuracy: 0.3187 - top-5-accuracy: 0.6719 - val_loss: 2.2061 - val_accuracy:
0.2587 - val_top-5-accuracy: 0.6289
Epoch 49/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1258 -
accuracy: 0.3196 - top-5-accuracy: 0.6653 - val_loss: 2.1823 - val_accuracy:
0.2804 - val_top-5-accuracy: 0.6433
Epoch 50/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1225 -
accuracy: 0.3200 - top-5-accuracy: 0.6694 - val_loss: 2.1641 - val_accuracy:
0.2918 - val_top-5-accuracy: 0.6458
Epoch 51/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1200 -
accuracy: 0.3217 - top-5-accuracy: 0.6707 - val_loss: 2.1740 - val_accuracy:
0.2849 - val_top-5-accuracy: 0.6529
Epoch 52/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1188 -
accuracy: 0.3235 - top-5-accuracy: 0.6700 - val_loss: 2.2034 - val_accuracy:
0.2578 - val_top-5-accuracy: 0.6344
Epoch 53/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1158 -
accuracy: 0.3254 - top-5-accuracy: 0.6720 - val_loss: 2.1964 - val_accuracy:
0.2762 - val_top-5-accuracy: 0.6438
```

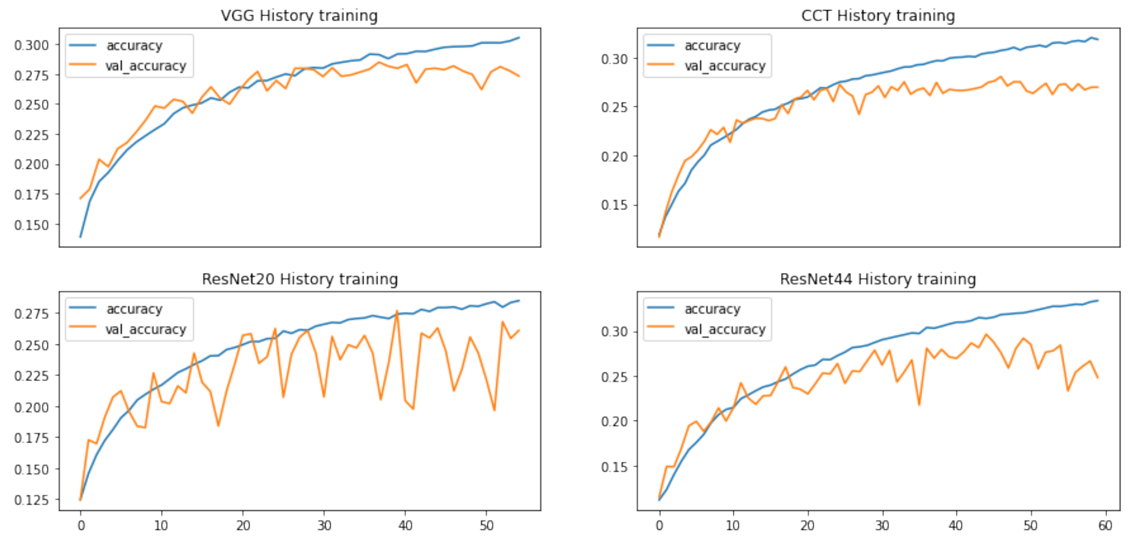
```
Epoch 54/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1152 -
accuracy: 0.3274 - top-5-accuracy: 0.6733 - val_loss: 2.1756 - val_accuracy:
0.2778 - val_top-5-accuracy: 0.6462
Epoch 55/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1133 -
accuracy: 0.3273 - top-5-accuracy: 0.6760 - val_loss: 2.1772 - val_accuracy:
0.2840 - val_top-5-accuracy: 0.6398
Epoch 56/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1127 -
accuracy: 0.3284 - top-5-accuracy: 0.6745 - val_loss: 2.2239 - val_accuracy:
0.2331 - val_top-5-accuracy: 0.6349
Epoch 57/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1106 -
accuracy: 0.3296 - top-5-accuracy: 0.6742 - val_loss: 2.2039 - val_accuracy:
0.2536 - val_top-5-accuracy: 0.6311
Epoch 58/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1099 -
accuracy: 0.3293 - top-5-accuracy: 0.6727 - val_loss: 2.2025 - val_accuracy:
0.2607 - val_top-5-accuracy: 0.6336
Epoch 59/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1053 -
accuracy: 0.3322 - top-5-accuracy: 0.6755 - val_loss: 2.1978 - val_accuracy:
0.2664 - val_top-5-accuracy: 0.6416
Epoch 60/100
317/317 [==============================] - 21s 67ms/step - loss: 2.1046 -
accuracy: 0.3336 - top-5-accuracy: 0.6762 - val_loss: 2.2151 - val_accuracy:
0.2480 - val_top-5-accuracy: 0.6067
157/157 [==============================] - 2s 10ms/step - loss: 1.6337 -
accuracy: 0.6724 - top-5-accuracy: 0.9572
Test accuracy: 67.24%
Test top 5 accuracy: 95.72%
```

Training history of our models :

```
[92]: plt.rcParams["figure.figsize"] = (15,7)
fig, axs = plt.subplots(2,2)
pd.DataFrame(history_vgg_classifier.history)[["accuracy","val_accuracy"]].
 ↪plot(ax = axs[0,0])
axs[0,0].set_title("VGG History training")
axs[0,0].get_xaxis().set_ticks([])
pd.DataFrame(history_cct_classifier.history)[["accuracy","val_accuracy"]].
 ↪plot(ax=axs[0,1])
axs[0,1].set_title("CCT History training")
axs[0,1].get_xaxis().set_ticks([])
pd.DataFrame(history_resnet20_classifier.history)[["accuracy","val_accuracy"]].
 ↪plot(ax=axs[1,0])
axs[1,0].set_title("ResNet20 History training")
```

```
pd.DataFrame(history_resnet44_classifier.history)[["accuracy","val_accuracy"]].
 ↪plot(ax = axs[1,1])
axs[1,1].set_title("ResNet44 History training")
```

```
[92]: Text(0.5, 1.0, 'ResNet44 History training')
```



Accruacy comparaison between the model on the test set

Model	Accuracy	#Params	Time per epoch
VGG	66.04%	0.8M	5s
CCT	62.02%	0.63M	9s
ResNet20	63.7%	0.10M	5s
ResNet44	67.24%	0.69M	21s

We can see from this table that the VGG model reach a good accuracy with a low computational time but a high memory cost.

1.7.1 2.3. Model II

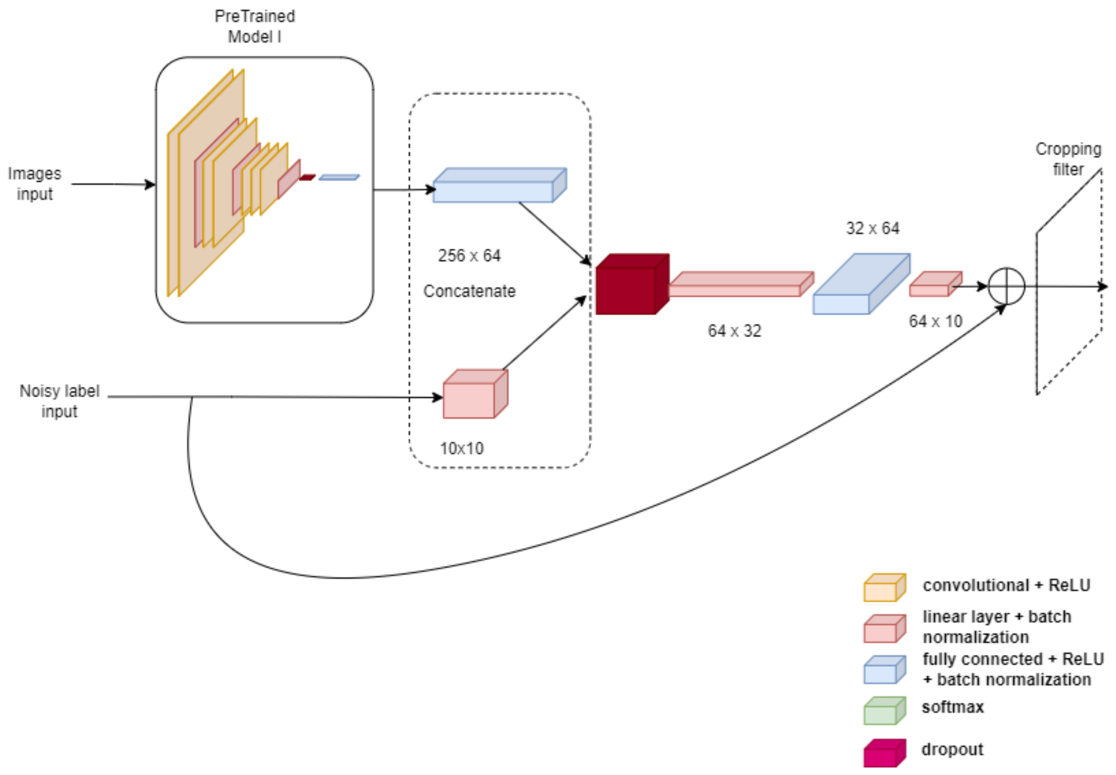
Create a label cleaning network

To improve our model accuracy on our training set, we are going to train a deep neural network as in Multi-Label Fashion Image Classification with Minimal Human Supervision that will relabel our noisy label and train our Model I on the new labels. Similarly to Inoue et al.(2017), we decide to decompose the training into two phases : 1. Training a label cleaning network with the cleaned labels 2. Train our Model I on the new labels

We ended up with the following architecture :

```
[4]: Image(filename="../figs/vgg16_xml.drawio.png")
```

[4]:



If Inoue et al.(2017) decided to replace every linear layer by a ReLU + BatchNormalization layers, we have found that keeping some linear layers + BatchNormalization worked better for this case.

For training this network, we separated our clean labels into a training set and a test set. The first 9000 images and clean labels were used for training and the last 1000 images and labels were used for testing. Again we use a batchsize of 128 and the same learning rate. We also use our pretrained `Model I` on the noisy label as our initializer of the feature extractor.

```
[21]: train_imgs_cleaning = imgs_prep[:9000]
      noisy_train_labels = smooth_labels(keras.utils.to_categorical(noisy_labels[:
       ↪9000]))
      cleaned_train_labels = smooth_labels(keras.utils.to_categorical(clean_labels[:
       ↪9000]))
      test_imgs_cleaning = imgs_prep[9000:10000]
      noisy_test_labels = smooth_labels(keras.utils.to_categorical(noisy_labels[9000:
       ↪10000]))
      cleaned_test_labels = smooth_labels(keras.utils.
       ↪to_categorical(clean_labels[9000:]))
```

```
[22]: def create_label_cleaner_clipped(model):
          image_inputs = model.input
          representation = layers.Dropout(0.2)(model.layers[-2].output)
```

```
    representation = layers.
↪Dense(64,activation="relu",kernel_initializer="he_uniform")(representation)
    representation = layers.BatchNormalization(trainable=True)(representation)
    # Create the Keras model.
    base_CNN = keras.Model(inputs=image_inputs, outputs=representation)
    label_input = layers.Input(shape=(10,))
    label_input = layers.Dense(10, activation = "linear")(label_input)
    combined_inputs = layers.concatenate([base_CNN.output,label_input])
    combined_inputs = layers.Dropout(0.5)(combined_inputs)
    label_cleaning_network = layers.
↪Dense(32,activation="linear",kernel_initializer='he_uniform')(combined_inputs)
    label_cleaning_network = layers.
↪BatchNormalization(trainable=True)(label_cleaning_network)
    label_cleaning_network = layers.
↪Dense(64,activation="relu",kernel_initializer='he_uniform')(combined_inputs)
    label_cleaning_network = layers.
↪BatchNormalization(trainable=True)(label_cleaning_network)
    label_cleaning_network = layers.
↪Dense(10,activation="linear",kernel_initializer='he_uniform')(label_cleaning_network)
    label_cleaning_network = layers.
↪BatchNormalization(trainable=True)(label_cleaning_network)
    label_cleaning_network = layers.Add()([label_input,label_cleaning_network])
    label_cleaning_network = Cliper()(label_cleaning_network)
    model = keras.Model(inputs=[base_CNN.input,label_input],␣
↪outputs=label_cleaning_network)
    return model
```

Train the label cleaning network (Multi-Label Fashion Image Classification with Minimal Human Supervision)

[23]:
```
batch_size = 128
num_epoch = 150
def run_cleaning_label(model):
    optimizer = tf.optimizers.Adam(
        learning_rate=learning_rate
    )
    model.compile(
        optimizer=optimizer,
        loss=keras.losses.CategoricalCrossentropy(from_logits=True),
        metrics=[
            keras.metrics.CategoricalAccuracy(name="accuracy"),
            keras.metrics.TopKCategoricalAccuracy(5, name="top-5-accuracy"),
        ],
    )
    checkpoint_filepath = "./tmp/checkpoint_"+str(model.name)
    checkpoint_callback = keras.callbacks.ModelCheckpoint(
        checkpoint_filepath,
        monitor="val_accuracy",
        save_best_only=True,
        save_weights_only=True,
    )
    early_stoping_callback = keras.callbacks.
↪EarlyStopping(monitor='val_accuracy', patience=15)

    history = model.fit(
        x=[train_imgs_cleaning,noisy_train_labels],
        y=cleaned_train_labels,
        batch_size=batch_size,
        epochs=num_epochs,
        validation_split=0.1,
        callbacks=[checkpoint_callback,early_stoping_callback],
    )
    model.load_weights(checkpoint_filepath)
    _, accuracy, _ = model.evaluate([test_imgs_cleaning,noisy_test_labels],␣
↪cleaned_test_labels)
    print(f"Test accuracy: {round(accuracy * 100, 2)}%")

    return history
```

[51]:

```
cleaning_network = create_label_cleaner_clipped(vgg_classifier)
cleaning_network._name = "cleaning_network_vgg_clipped"
history_cleaning_vgg = run_cleaning_label(cleaning_network)
```

```
Epoch 1/100
64/64 [==============================] - 4s 29ms/step - loss: 2.1012 - accuracy:
0.3002 - top-5-accuracy: 0.8337 - val_loss: 1.9119 - val_accuracy: 0.5456 -
val_top-5-accuracy: 0.9367
Epoch 2/100
64/64 [==============================] - 1s 17ms/step - loss: 1.9256 - accuracy:
0.4473 - top-5-accuracy: 0.9322 - val_loss: 1.8535 - val_accuracy: 0.5967 -
val_top-5-accuracy: 0.9489
Epoch 3/100
64/64 [==============================] - 1s 16ms/step - loss: 1.8804 - accuracy:
0.5014 - top-5-accuracy: 0.9502 - val_loss: 1.8420 - val_accuracy: 0.5667 -
val_top-5-accuracy: 0.9456
Epoch 4/100
64/64 [==============================] - 1s 17ms/step - loss: 1.8524 - accuracy:
0.5495 - top-5-accuracy: 0.9558 - val_loss: 1.8210 - val_accuracy: 0.6089 -
val_top-5-accuracy: 0.9689
Epoch 5/100
64/64 [==============================] - 1s 20ms/step - loss: 1.8328 - accuracy:
0.5800 - top-5-accuracy: 0.9601 - val_loss: 1.8106 - val_accuracy: 0.6178 -
val_top-5-accuracy: 0.9722
Epoch 6/100
64/64 [==============================] - 1s 16ms/step - loss: 1.8208 - accuracy:
```

```
0.6002 - top-5-accuracy: 0.9706 - val_loss: 1.8024 - val_accuracy: 0.6178 -
val_top-5-accuracy: 0.9767
Epoch 7/100
64/64 [==============================] - 1s 16ms/step - loss: 1.8099 - accuracy:
0.6191 - top-5-accuracy: 0.9737 - val_loss: 1.8041 - val_accuracy: 0.5978 -
val_top-5-accuracy: 0.9811
Epoch 8/100
64/64 [==============================] - 1s 17ms/step - loss: 1.8024 - accuracy:
0.6290 - top-5-accuracy: 0.9770 - val_loss: 1.7824 - val_accuracy: 0.6444 -
val_top-5-accuracy: 0.9789
Epoch 9/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7977 - accuracy:
0.6384 - top-5-accuracy: 0.9768 - val_loss: 1.7995 - val_accuracy: 0.6344 -
val_top-5-accuracy: 0.9844
Epoch 10/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7863 - accuracy:
0.6432 - top-5-accuracy: 0.9816 - val_loss: 1.7892 - val_accuracy: 0.6333 -
val_top-5-accuracy: 0.9856
Epoch 11/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7832 - accuracy:
0.6551 - top-5-accuracy: 0.9835 - val_loss: 1.7885 - val_accuracy: 0.6522 -
val_top-5-accuracy: 0.9844
Epoch 12/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7835 - accuracy:
0.6560 - top-5-accuracy: 0.9819 - val_loss: 1.8139 - val_accuracy: 0.6244 -
val_top-5-accuracy: 0.9856
Epoch 13/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7732 - accuracy:
0.6667 - top-5-accuracy: 0.9838 - val_loss: 1.7729 - val_accuracy: 0.6667 -
val_top-5-accuracy: 0.9844
Epoch 14/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7745 - accuracy:
0.6652 - top-5-accuracy: 0.9842 - val_loss: 1.7767 - val_accuracy: 0.6633 -
val_top-5-accuracy: 0.9844
Epoch 15/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7737 - accuracy:
0.6715 - top-5-accuracy: 0.9837 - val_loss: 1.7644 - val_accuracy: 0.6800 -
val_top-5-accuracy: 0.9889
Epoch 16/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7626 - accuracy:
0.6819 - top-5-accuracy: 0.9872 - val_loss: 1.7735 - val_accuracy: 0.6744 -
val_top-5-accuracy: 0.9922
Epoch 17/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7616 - accuracy:
0.6896 - top-5-accuracy: 0.9884 - val_loss: 1.7530 - val_accuracy: 0.6911 -
val_top-5-accuracy: 0.9889
Epoch 18/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7552 - accuracy:
```

```
0.6923 - top-5-accuracy: 0.9872 - val_loss: 1.7660 - val_accuracy: 0.6911 -
val_top-5-accuracy: 0.9878
Epoch 19/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7558 - accuracy:
0.6949 - top-5-accuracy: 0.9870 - val_loss: 1.7696 - val_accuracy: 0.6778 -
val_top-5-accuracy: 0.9900
Epoch 20/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7534 - accuracy:
0.6995 - top-5-accuracy: 0.9894 - val_loss: 1.7727 - val_accuracy: 0.6689 -
val_top-5-accuracy: 0.9867
Epoch 21/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7497 - accuracy:
0.7051 - top-5-accuracy: 0.9875 - val_loss: 1.7878 - val_accuracy: 0.6833 -
val_top-5-accuracy: 0.9911
Epoch 22/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7397 - accuracy:
0.7151 - top-5-accuracy: 0.9902 - val_loss: 1.7592 - val_accuracy: 0.6911 -
val_top-5-accuracy: 0.9944
Epoch 23/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7443 - accuracy:
0.7147 - top-5-accuracy: 0.9917 - val_loss: 1.7907 - val_accuracy: 0.6567 -
val_top-5-accuracy: 0.9867
Epoch 24/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7387 - accuracy:
0.7252 - top-5-accuracy: 0.9905 - val_loss: 1.7686 - val_accuracy: 0.6789 -
val_top-5-accuracy: 0.9878
Epoch 25/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7320 - accuracy:
0.7309 - top-5-accuracy: 0.9916 - val_loss: 1.7525 - val_accuracy: 0.6811 -
val_top-5-accuracy: 0.9944
Epoch 26/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7288 - accuracy:
0.7357 - top-5-accuracy: 0.9917 - val_loss: 1.7698 - val_accuracy: 0.6833 -
val_top-5-accuracy: 0.9889
Epoch 27/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7349 - accuracy:
0.7214 - top-5-accuracy: 0.9930 - val_loss: 1.7488 - val_accuracy: 0.6989 -
val_top-5-accuracy: 0.9933
Epoch 28/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7346 - accuracy:
0.7244 - top-5-accuracy: 0.9928 - val_loss: 1.7462 - val_accuracy: 0.7022 -
val_top-5-accuracy: 0.9867
Epoch 29/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7315 - accuracy:
0.7321 - top-5-accuracy: 0.9912 - val_loss: 1.7550 - val_accuracy: 0.6978 -
val_top-5-accuracy: 0.9933
Epoch 30/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7244 - accuracy:
```

```
0.7346 - top-5-accuracy: 0.9944 - val_loss: 1.7516 - val_accuracy: 0.6956 -
val_top-5-accuracy: 0.9956
Epoch 31/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7233 - accuracy:
0.7419 - top-5-accuracy: 0.9944 - val_loss: 1.7349 - val_accuracy: 0.7200 -
val_top-5-accuracy: 0.9933
Epoch 32/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7228 - accuracy:
0.7404 - top-5-accuracy: 0.9935 - val_loss: 1.7579 - val_accuracy: 0.6956 -
val_top-5-accuracy: 0.9933
Epoch 33/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7246 - accuracy:
0.7364 - top-5-accuracy: 0.9946 - val_loss: 1.7518 - val_accuracy: 0.7089 -
val_top-5-accuracy: 0.9944
Epoch 34/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7185 - accuracy:
0.7478 - top-5-accuracy: 0.9952 - val_loss: 1.7604 - val_accuracy: 0.7111 -
val_top-5-accuracy: 0.9933
Epoch 35/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7239 - accuracy:
0.7422 - top-5-accuracy: 0.9926 - val_loss: 1.7474 - val_accuracy: 0.7200 -
val_top-5-accuracy: 0.9967
Epoch 36/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7111 - accuracy:
0.7574 - top-5-accuracy: 0.9948 - val_loss: 1.7539 - val_accuracy: 0.7067 -
val_top-5-accuracy: 0.9967
Epoch 37/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7120 - accuracy:
0.7574 - top-5-accuracy: 0.9948 - val_loss: 1.7439 - val_accuracy: 0.7089 -
val_top-5-accuracy: 0.9956
Epoch 38/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7179 - accuracy:
0.7438 - top-5-accuracy: 0.9944 - val_loss: 1.7484 - val_accuracy: 0.6911 -
val_top-5-accuracy: 0.9911
Epoch 39/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7112 - accuracy:
0.7565 - top-5-accuracy: 0.9944 - val_loss: 1.7416 - val_accuracy: 0.7322 -
val_top-5-accuracy: 0.9933
Epoch 40/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7065 - accuracy:
0.7652 - top-5-accuracy: 0.9964 - val_loss: 1.7593 - val_accuracy: 0.7067 -
val_top-5-accuracy: 0.9933
Epoch 41/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7091 - accuracy:
0.7605 - top-5-accuracy: 0.9956 - val_loss: 1.7491 - val_accuracy: 0.7167 -
val_top-5-accuracy: 0.9944
Epoch 42/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7076 - accuracy:
```

```
0.7636 - top-5-accuracy: 0.9946 - val_loss: 1.7492 - val_accuracy: 0.7111 -
val_top-5-accuracy: 0.9900
Epoch 43/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7050 - accuracy:
0.7623 - top-5-accuracy: 0.9967 - val_loss: 1.7507 - val_accuracy: 0.7022 -
val_top-5-accuracy: 0.9956
Epoch 44/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7045 - accuracy:
0.7716 - top-5-accuracy: 0.9958 - val_loss: 1.7435 - val_accuracy: 0.7200 -
val_top-5-accuracy: 0.9967
Epoch 45/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7034 - accuracy:
0.7714 - top-5-accuracy: 0.9962 - val_loss: 1.7510 - val_accuracy: 0.7033 -
val_top-5-accuracy: 0.9944
Epoch 46/100
64/64 [==============================] - 1s 16ms/step - loss: 1.7006 - accuracy:
0.7717 - top-5-accuracy: 0.9960 - val_loss: 1.7602 - val_accuracy: 0.6800 -
val_top-5-accuracy: 0.9911
Epoch 47/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6985 - accuracy:
0.7768 - top-5-accuracy: 0.9959 - val_loss: 1.7299 - val_accuracy: 0.7244 -
val_top-5-accuracy: 0.9911
Epoch 48/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6938 - accuracy:
0.7852 - top-5-accuracy: 0.9974 - val_loss: 1.7331 - val_accuracy: 0.7244 -
val_top-5-accuracy: 0.9933
Epoch 49/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6955 - accuracy:
0.7780 - top-5-accuracy: 0.9977 - val_loss: 1.7304 - val_accuracy: 0.7289 -
val_top-5-accuracy: 0.9933
Epoch 50/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6934 - accuracy:
0.7783 - top-5-accuracy: 0.9974 - val_loss: 1.7512 - val_accuracy: 0.7244 -
val_top-5-accuracy: 0.9956
Epoch 51/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6970 - accuracy:
0.7778 - top-5-accuracy: 0.9967 - val_loss: 1.7468 - val_accuracy: 0.7211 -
val_top-5-accuracy: 0.9944
Epoch 52/100
64/64 [==============================] - 1s 18ms/step - loss: 1.6907 - accuracy:
0.7895 - top-5-accuracy: 0.9972 - val_loss: 1.7264 - val_accuracy: 0.7411 -
val_top-5-accuracy: 0.9989
Epoch 53/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6993 - accuracy:
0.7751 - top-5-accuracy: 0.9964 - val_loss: 1.7368 - val_accuracy: 0.7278 -
val_top-5-accuracy: 0.9911
Epoch 54/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6901 - accuracy:
```

```
0.7904 - top-5-accuracy: 0.9975 - val_loss: 1.7429 - val_accuracy: 0.7089 -
val_top-5-accuracy: 0.9956
Epoch 55/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6888 - accuracy:
0.7851 - top-5-accuracy: 0.9973 - val_loss: 1.7376 - val_accuracy: 0.7244 -
val_top-5-accuracy: 0.9944
Epoch 56/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6888 - accuracy:
0.7951 - top-5-accuracy: 0.9970 - val_loss: 1.7409 - val_accuracy: 0.7156 -
val_top-5-accuracy: 0.9933
Epoch 57/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6883 - accuracy:
0.7841 - top-5-accuracy: 0.9967 - val_loss: 1.7455 - val_accuracy: 0.7211 -
val_top-5-accuracy: 0.9944
Epoch 58/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6878 - accuracy:
0.7893 - top-5-accuracy: 0.9964 - val_loss: 1.7301 - val_accuracy: 0.7278 -
val_top-5-accuracy: 0.9978
Epoch 59/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6897 - accuracy:
0.7904 - top-5-accuracy: 0.9973 - val_loss: 1.7245 - val_accuracy: 0.7400 -
val_top-5-accuracy: 0.9933
Epoch 60/100
64/64 [==============================] - 1s 17ms/step - loss: 1.6820 - accuracy:
0.8011 - top-5-accuracy: 0.9977 - val_loss: 1.7297 - val_accuracy: 0.7422 -
val_top-5-accuracy: 0.9956
Epoch 61/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6830 - accuracy:
0.8025 - top-5-accuracy: 0.9968 - val_loss: 1.7442 - val_accuracy: 0.7167 -
val_top-5-accuracy: 0.9956
Epoch 62/100
64/64 [==============================] - 1s 17ms/step - loss: 1.6762 - accuracy:
0.8083 - top-5-accuracy: 0.9986 - val_loss: 1.7281 - val_accuracy: 0.7456 -
val_top-5-accuracy: 0.9978
Epoch 63/100
64/64 [==============================] - 1s 19ms/step - loss: 1.6779 - accuracy:
0.8072 - top-5-accuracy: 0.9979 - val_loss: 1.7375 - val_accuracy: 0.7300 -
val_top-5-accuracy: 0.9967
Epoch 64/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6783 - accuracy:
0.8010 - top-5-accuracy: 0.9990 - val_loss: 1.7439 - val_accuracy: 0.7056 -
val_top-5-accuracy: 0.9922
Epoch 65/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6796 - accuracy:
0.8048 - top-5-accuracy: 0.9970 - val_loss: 1.7241 - val_accuracy: 0.7289 -
val_top-5-accuracy: 0.9967
Epoch 66/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6797 - accuracy:
```

```
0.8054 - top-5-accuracy: 0.9978 - val_loss: 1.7354 - val_accuracy: 0.7422 -
val_top-5-accuracy: 0.9967
Epoch 67/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6734 - accuracy:
0.8091 - top-5-accuracy: 0.9969 - val_loss: 1.7327 - val_accuracy: 0.7456 -
val_top-5-accuracy: 0.9967
Epoch 68/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6773 - accuracy:
0.8078 - top-5-accuracy: 0.9979 - val_loss: 1.7301 - val_accuracy: 0.7300 -
val_top-5-accuracy: 0.9978
Epoch 69/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6757 - accuracy:
0.8090 - top-5-accuracy: 0.9983 - val_loss: 1.7286 - val_accuracy: 0.7178 -
val_top-5-accuracy: 0.9967
Epoch 70/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6735 - accuracy:
0.8136 - top-5-accuracy: 0.9979 - val_loss: 1.7360 - val_accuracy: 0.7411 -
val_top-5-accuracy: 0.9978
Epoch 71/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6743 - accuracy:
0.8167 - top-5-accuracy: 0.9980 - val_loss: 1.7489 - val_accuracy: 0.7222 -
val_top-5-accuracy: 0.9989
Epoch 72/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6721 - accuracy:
0.8159 - top-5-accuracy: 0.9980 - val_loss: 1.7390 - val_accuracy: 0.7322 -
val_top-5-accuracy: 0.9978
Epoch 73/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6703 - accuracy:
0.8168 - top-5-accuracy: 0.9975 - val_loss: 1.7294 - val_accuracy: 0.7422 -
val_top-5-accuracy: 0.9967
Epoch 74/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6747 - accuracy:
0.8090 - top-5-accuracy: 0.9973 - val_loss: 1.7431 - val_accuracy: 0.7189 -
val_top-5-accuracy: 0.9956
Epoch 75/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6712 - accuracy:
0.8151 - top-5-accuracy: 0.9988 - val_loss: 1.7535 - val_accuracy: 0.6989 -
val_top-5-accuracy: 0.9978
Epoch 76/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6719 - accuracy:
0.8135 - top-5-accuracy: 0.9985 - val_loss: 1.7366 - val_accuracy: 0.7322 -
val_top-5-accuracy: 0.9978
Epoch 77/100
64/64 [==============================] - 1s 16ms/step - loss: 1.6687 - accuracy:
0.8162 - top-5-accuracy: 0.9978 - val_loss: 1.7268 - val_accuracy: 0.7444 -
val_top-5-accuracy: 0.9967
32/32 [==============================] - 0s 4ms/step - loss: 1.7223 - accuracy:
0.7660 - top-5-accuracy: 0.9960
```

```
Test accuracy: 76.6%
```

[52]:
```
cleaning_network = create_label_cleaner_clipped(cct_classifier)
cleaning_network._name = "cleaning_network_cct_clipped"
history_cleaning_cct = run_cleaning_label(cleaning_network)
```

```
Epoch 1/100
64/64 [==============================] - 7s 45ms/step - loss: 2.0771 - accuracy:
0.3293 - top-5-accuracy: 0.8552 - val_loss: 1.8976 - val_accuracy: 0.5444 -
val_top-5-accuracy: 0.9267
Epoch 2/100
64/64 [==============================] - 2s 30ms/step - loss: 1.9097 - accuracy:
0.4731 - top-5-accuracy: 0.9375 - val_loss: 1.8502 - val_accuracy: 0.6067 -
val_top-5-accuracy: 0.9511
Epoch 3/100
64/64 [==============================] - 2s 29ms/step - loss: 1.8564 - accuracy:
0.5312 - top-5-accuracy: 0.9607 - val_loss: 1.8215 - val_accuracy: 0.5989 -
val_top-5-accuracy: 0.9656
Epoch 4/100
64/64 [==============================] - 2s 30ms/step - loss: 1.8353 - accuracy:
0.5746 - top-5-accuracy: 0.9694 - val_loss: 1.8045 - val_accuracy: 0.6256 -
val_top-5-accuracy: 0.9833
Epoch 5/100
64/64 [==============================] - 2s 30ms/step - loss: 1.8196 - accuracy:
0.5954 - top-5-accuracy: 0.9770 - val_loss: 1.7993 - val_accuracy: 0.6500 -
val_top-5-accuracy: 0.9756
Epoch 6/100
64/64 [==============================] - 2s 30ms/step - loss: 1.8056 - accuracy:
0.6084 - top-5-accuracy: 0.9794 - val_loss: 1.7818 - val_accuracy: 0.6511 -
val_top-5-accuracy: 0.9856
Epoch 7/100
64/64 [==============================] - 2s 28ms/step - loss: 1.7970 - accuracy:
0.6264 - top-5-accuracy: 0.9814 - val_loss: 1.7886 - val_accuracy: 0.6344 -
val_top-5-accuracy: 0.9867
Epoch 8/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7846 - accuracy:
0.6438 - top-5-accuracy: 0.9836 - val_loss: 1.7977 - val_accuracy: 0.6111 -
val_top-5-accuracy: 0.9844
Epoch 9/100
64/64 [==============================] - 2s 30ms/step - loss: 1.7825 - accuracy:
0.6470 - top-5-accuracy: 0.9868 - val_loss: 1.7807 - val_accuracy: 0.6678 -
val_top-5-accuracy: 0.9822
Epoch 10/100
64/64 [==============================] - 2s 28ms/step - loss: 1.7777 - accuracy:
0.6546 - top-5-accuracy: 0.9869 - val_loss: 1.7911 - val_accuracy: 0.6478 -
val_top-5-accuracy: 0.9878
Epoch 11/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7718 - accuracy:
```

```
0.6642 - top-5-accuracy: 0.9868 - val_loss: 1.7778 - val_accuracy: 0.6644 -
val_top-5-accuracy: 0.9922
Epoch 12/100
64/64 [==============================] - 2s 28ms/step - loss: 1.7682 - accuracy:
0.6626 - top-5-accuracy: 0.9899 - val_loss: 1.7923 - val_accuracy: 0.6344 -
val_top-5-accuracy: 0.9944
Epoch 13/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7581 - accuracy:
0.6825 - top-5-accuracy: 0.9896 - val_loss: 1.7851 - val_accuracy: 0.6533 -
val_top-5-accuracy: 0.9900
Epoch 14/100
64/64 [==============================] - 2s 28ms/step - loss: 1.7585 - accuracy:
0.6822 - top-5-accuracy: 0.9906 - val_loss: 1.7811 - val_accuracy: 0.6622 -
val_top-5-accuracy: 0.9911
Epoch 15/100
64/64 [==============================] - 2s 30ms/step - loss: 1.7525 - accuracy:
0.6883 - top-5-accuracy: 0.9904 - val_loss: 1.7735 - val_accuracy: 0.6933 -
val_top-5-accuracy: 0.9922
Epoch 16/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7523 - accuracy:
0.6936 - top-5-accuracy: 0.9931 - val_loss: 1.7793 - val_accuracy: 0.6900 -
val_top-5-accuracy: 0.9922
Epoch 17/100
64/64 [==============================] - 2s 28ms/step - loss: 1.7435 - accuracy:
0.7030 - top-5-accuracy: 0.9919 - val_loss: 1.7523 - val_accuracy: 0.6911 -
val_top-5-accuracy: 0.9956
Epoch 18/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7398 - accuracy:
0.7043 - top-5-accuracy: 0.9923 - val_loss: 1.7778 - val_accuracy: 0.6600 -
val_top-5-accuracy: 0.9922
Epoch 19/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7429 - accuracy:
0.7086 - top-5-accuracy: 0.9926 - val_loss: 1.7934 - val_accuracy: 0.6700 -
val_top-5-accuracy: 0.9933
Epoch 20/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7368 - accuracy:
0.7142 - top-5-accuracy: 0.9921 - val_loss: 1.8017 - val_accuracy: 0.6422 -
val_top-5-accuracy: 0.9933
Epoch 21/100
64/64 [==============================] - 2s 31ms/step - loss: 1.7312 - accuracy:
0.7257 - top-5-accuracy: 0.9940 - val_loss: 1.7766 - val_accuracy: 0.6822 -
val_top-5-accuracy: 0.9889
Epoch 22/100
64/64 [==============================] - 2s 30ms/step - loss: 1.7289 - accuracy:
0.7259 - top-5-accuracy: 0.9954 - val_loss: 1.7574 - val_accuracy: 0.6978 -
val_top-5-accuracy: 0.9944
Epoch 23/100
64/64 [==============================] - 2s 28ms/step - loss: 1.7247 - accuracy:
```

```
0.7380 - top-5-accuracy: 0.9943 - val_loss: 1.7663 - val_accuracy: 0.6856 -
val_top-5-accuracy: 0.9933
Epoch 24/100
64/64 [==============================] - 2s 30ms/step - loss: 1.7261 - accuracy:
0.7373 - top-5-accuracy: 0.9951 - val_loss: 1.7687 - val_accuracy: 0.7011 -
val_top-5-accuracy: 0.9967
Epoch 25/100
64/64 [==============================] - 2s 30ms/step - loss: 1.7187 - accuracy:
0.7435 - top-5-accuracy: 0.9956 - val_loss: 1.7384 - val_accuracy: 0.7322 -
val_top-5-accuracy: 0.9978
Epoch 26/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7156 - accuracy:
0.7432 - top-5-accuracy: 0.9962 - val_loss: 1.7616 - val_accuracy: 0.6844 -
val_top-5-accuracy: 0.9922
Epoch 27/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7150 - accuracy:
0.7462 - top-5-accuracy: 0.9962 - val_loss: 1.7540 - val_accuracy: 0.6811 -
val_top-5-accuracy: 0.9967
Epoch 28/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7117 - accuracy:
0.7526 - top-5-accuracy: 0.9956 - val_loss: 1.7674 - val_accuracy: 0.7100 -
val_top-5-accuracy: 0.9944
Epoch 29/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7113 - accuracy:
0.7567 - top-5-accuracy: 0.9957 - val_loss: 1.7661 - val_accuracy: 0.6978 -
val_top-5-accuracy: 0.9978
Epoch 30/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7091 - accuracy:
0.7611 - top-5-accuracy: 0.9970 - val_loss: 1.7678 - val_accuracy: 0.6767 -
val_top-5-accuracy: 0.9933
Epoch 31/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7111 - accuracy:
0.7557 - top-5-accuracy: 0.9963 - val_loss: 1.7513 - val_accuracy: 0.6733 -
val_top-5-accuracy: 0.9989
Epoch 32/100
64/64 [==============================] - 2s 29ms/step - loss: 1.7037 - accuracy:
0.7657 - top-5-accuracy: 0.9962 - val_loss: 1.7461 - val_accuracy: 0.7078 -
val_top-5-accuracy: 0.9967
Epoch 33/100
64/64 [==============================] - 2s 28ms/step - loss: 1.7000 - accuracy:
0.7707 - top-5-accuracy: 0.9969 - val_loss: 1.7848 - val_accuracy: 0.6711 -
val_top-5-accuracy: 0.9956
Epoch 34/100
64/64 [==============================] - 2s 29ms/step - loss: 1.6997 - accuracy:
0.7730 - top-5-accuracy: 0.9962 - val_loss: 1.7662 - val_accuracy: 0.6944 -
val_top-5-accuracy: 0.9967
Epoch 35/100
64/64 [==============================] - 2s 29ms/step - loss: 1.6964 - accuracy:
```

```
0.7790 - top-5-accuracy: 0.9970 - val_loss: 1.7471 - val_accuracy: 0.7022 -
val_top-5-accuracy: 0.9978
Epoch 36/100
64/64 [==============================] - 2s 29ms/step - loss: 1.6899 - accuracy:
0.7851 - top-5-accuracy: 0.9968 - val_loss: 1.7427 - val_accuracy: 0.7167 -
val_top-5-accuracy: 0.9944
Epoch 37/100
64/64 [==============================] - 2s 28ms/step - loss: 1.6925 - accuracy:
0.7894 - top-5-accuracy: 0.9972 - val_loss: 1.7537 - val_accuracy: 0.6933 -
val_top-5-accuracy: 0.9989
Epoch 38/100
64/64 [==============================] - 2s 28ms/step - loss: 1.6891 - accuracy:
0.7910 - top-5-accuracy: 0.9977 - val_loss: 1.7634 - val_accuracy: 0.7022 -
val_top-5-accuracy: 0.9922
Epoch 39/100
64/64 [==============================] - 2s 29ms/step - loss: 1.6897 - accuracy:
0.7899 - top-5-accuracy: 0.9986 - val_loss: 1.7652 - val_accuracy: 0.6933 -
val_top-5-accuracy: 0.9922
Epoch 40/100
64/64 [==============================] - 2s 28ms/step - loss: 1.6869 - accuracy:
0.7984 - top-5-accuracy: 0.9980 - val_loss: 1.7542 - val_accuracy: 0.7056 -
val_top-5-accuracy: 0.9956
32/32 [==============================] - 0s 7ms/step - loss: 1.7461 - accuracy:
0.7050 - top-5-accuracy: 0.9900
Test accuracy: 70.5%
```

[53]:
```
cleaning_network = create_label_cleaner_clipped(resnet20_classifier)
cleaning_network._name = "cleaning_network_resnet20_clipped"
history_cleaning_resnet20 = run_cleaning_label(cleaning_network)
```

```
Epoch 1/100
64/64 [==============================] - 4s 30ms/step - loss: 2.0541 - accuracy:
0.3278 - top-5-accuracy: 0.8663 - val_loss: 1.9707 - val_accuracy: 0.4144 -
val_top-5-accuracy: 0.9189
Epoch 2/100
64/64 [==============================] - 1s 18ms/step - loss: 1.9258 - accuracy:
0.4306 - top-5-accuracy: 0.9332 - val_loss: 1.9939 - val_accuracy: 0.3956 -
val_top-5-accuracy: 0.9156
Epoch 3/100
64/64 [==============================] - 1s 18ms/step - loss: 1.8894 - accuracy:
0.4567 - top-5-accuracy: 0.9535 - val_loss: 1.8776 - val_accuracy: 0.4667 -
val_top-5-accuracy: 0.9556
Epoch 4/100
64/64 [==============================] - 1s 19ms/step - loss: 1.8645 - accuracy:
0.5036 - top-5-accuracy: 0.9553 - val_loss: 1.8624 - val_accuracy: 0.5111 -
val_top-5-accuracy: 0.9544
Epoch 5/100
64/64 [==============================] - 1s 19ms/step - loss: 1.8487 - accuracy:
```

```
0.5190 - top-5-accuracy: 0.9621 - val_loss: 1.8465 - val_accuracy: 0.5122 -
val_top-5-accuracy: 0.9644
Epoch 6/100
64/64 [==============================] - 1s 18ms/step - loss: 1.8363 - accuracy:
0.5407 - top-5-accuracy: 0.9707 - val_loss: 1.8925 - val_accuracy: 0.5378 -
val_top-5-accuracy: 0.9700
Epoch 7/100
64/64 [==============================] - 1s 17ms/step - loss: 1.8282 - accuracy:
0.5714 - top-5-accuracy: 0.9696 - val_loss: 1.9315 - val_accuracy: 0.4478 -
val_top-5-accuracy: 0.9722
Epoch 8/100
64/64 [==============================] - 1s 17ms/step - loss: 1.8250 - accuracy:
0.5741 - top-5-accuracy: 0.9737 - val_loss: 1.8899 - val_accuracy: 0.5111 -
val_top-5-accuracy: 0.9767
Epoch 9/100
64/64 [==============================] - 1s 18ms/step - loss: 1.8173 - accuracy:
0.5983 - top-5-accuracy: 0.9749 - val_loss: 1.8593 - val_accuracy: 0.5867 -
val_top-5-accuracy: 0.9778
Epoch 10/100
64/64 [==============================] - 1s 18ms/step - loss: 1.8079 - accuracy:
0.6130 - top-5-accuracy: 0.9793 - val_loss: 1.8186 - val_accuracy: 0.6078 -
val_top-5-accuracy: 0.9778
Epoch 11/100
64/64 [==============================] - 1s 17ms/step - loss: 1.8019 - accuracy:
0.6121 - top-5-accuracy: 0.9775 - val_loss: 1.8901 - val_accuracy: 0.5167 -
val_top-5-accuracy: 0.9778
Epoch 12/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7957 - accuracy:
0.6346 - top-5-accuracy: 0.9801 - val_loss: 1.8431 - val_accuracy: 0.6067 -
val_top-5-accuracy: 0.9778
Epoch 13/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7962 - accuracy:
0.6264 - top-5-accuracy: 0.9815 - val_loss: 1.8868 - val_accuracy: 0.5056 -
val_top-5-accuracy: 0.9889
Epoch 14/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7913 - accuracy:
0.6295 - top-5-accuracy: 0.9836 - val_loss: 1.8475 - val_accuracy: 0.5833 -
val_top-5-accuracy: 0.9822
Epoch 15/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7903 - accuracy:
0.6364 - top-5-accuracy: 0.9832 - val_loss: 1.8141 - val_accuracy: 0.6067 -
val_top-5-accuracy: 0.9844
Epoch 16/100
64/64 [==============================] - 1s 19ms/step - loss: 1.7896 - accuracy:
0.6317 - top-5-accuracy: 0.9867 - val_loss: 1.8261 - val_accuracy: 0.6178 -
val_top-5-accuracy: 0.9889
Epoch 17/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7843 - accuracy:
```

```
0.6457 - top-5-accuracy: 0.9844 - val_loss: 1.8204 - val_accuracy: 0.6000 -
val_top-5-accuracy: 0.9778
Epoch 18/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7819 - accuracy:
0.6432 - top-5-accuracy: 0.9853 - val_loss: 1.8052 - val_accuracy: 0.5844 -
val_top-5-accuracy: 0.9933
Epoch 19/100
64/64 [==============================] - 1s 21ms/step - loss: 1.7827 - accuracy:
0.6377 - top-5-accuracy: 0.9864 - val_loss: 1.8765 - val_accuracy: 0.5256 -
val_top-5-accuracy: 0.9878
Epoch 20/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7780 - accuracy:
0.6457 - top-5-accuracy: 0.9872 - val_loss: 1.9147 - val_accuracy: 0.5133 -
val_top-5-accuracy: 0.9867
Epoch 21/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7780 - accuracy:
0.6452 - top-5-accuracy: 0.9867 - val_loss: 1.8348 - val_accuracy: 0.5756 -
val_top-5-accuracy: 0.9900
Epoch 22/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7768 - accuracy:
0.6479 - top-5-accuracy: 0.9841 - val_loss: 1.8192 - val_accuracy: 0.6189 -
val_top-5-accuracy: 0.9933
Epoch 23/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7734 - accuracy:
0.6507 - top-5-accuracy: 0.9883 - val_loss: 1.8318 - val_accuracy: 0.6244 -
val_top-5-accuracy: 0.9856
Epoch 24/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7785 - accuracy:
0.6467 - top-5-accuracy: 0.9896 - val_loss: 1.8114 - val_accuracy: 0.6189 -
val_top-5-accuracy: 0.9911
Epoch 25/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7694 - accuracy:
0.6627 - top-5-accuracy: 0.9881 - val_loss: 1.8308 - val_accuracy: 0.6033 -
val_top-5-accuracy: 0.9856
Epoch 26/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7708 - accuracy:
0.6583 - top-5-accuracy: 0.9880 - val_loss: 1.8285 - val_accuracy: 0.6133 -
val_top-5-accuracy: 0.9922
Epoch 27/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7694 - accuracy:
0.6578 - top-5-accuracy: 0.9896 - val_loss: 1.7954 - val_accuracy: 0.6300 -
val_top-5-accuracy: 0.9867
Epoch 28/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7678 - accuracy:
0.6541 - top-5-accuracy: 0.9896 - val_loss: 1.7988 - val_accuracy: 0.6233 -
val_top-5-accuracy: 0.9878
Epoch 29/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7642 - accuracy:
```

```
0.6705 - top-5-accuracy: 0.9893 - val_loss: 1.8015 - val_accuracy: 0.6211 -
val_top-5-accuracy: 0.9922
Epoch 30/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7614 - accuracy:
0.6679 - top-5-accuracy: 0.9912 - val_loss: 1.8119 - val_accuracy: 0.6422 -
val_top-5-accuracy: 0.9889
Epoch 31/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7590 - accuracy:
0.6648 - top-5-accuracy: 0.9911 - val_loss: 1.7909 - val_accuracy: 0.6344 -
val_top-5-accuracy: 0.9911
Epoch 32/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7577 - accuracy:
0.6668 - top-5-accuracy: 0.9923 - val_loss: 1.7891 - val_accuracy: 0.6189 -
val_top-5-accuracy: 0.9911
Epoch 33/100
64/64 [==============================] - 1s 19ms/step - loss: 1.7544 - accuracy:
0.6749 - top-5-accuracy: 0.9898 - val_loss: 1.8029 - val_accuracy: 0.6556 -
val_top-5-accuracy: 0.9922
Epoch 34/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7580 - accuracy:
0.6774 - top-5-accuracy: 0.9909 - val_loss: 1.8049 - val_accuracy: 0.6344 -
val_top-5-accuracy: 0.9978
Epoch 35/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7534 - accuracy:
0.6785 - top-5-accuracy: 0.9912 - val_loss: 1.8429 - val_accuracy: 0.5711 -
val_top-5-accuracy: 0.9844
Epoch 36/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7552 - accuracy:
0.6846 - top-5-accuracy: 0.9912 - val_loss: 1.8061 - val_accuracy: 0.6278 -
val_top-5-accuracy: 0.9911
Epoch 37/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7553 - accuracy:
0.6786 - top-5-accuracy: 0.9919 - val_loss: 1.7999 - val_accuracy: 0.6056 -
val_top-5-accuracy: 0.9944
Epoch 38/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7495 - accuracy:
0.6894 - top-5-accuracy: 0.9915 - val_loss: 1.7851 - val_accuracy: 0.6522 -
val_top-5-accuracy: 0.9889
Epoch 39/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7456 - accuracy:
0.6877 - top-5-accuracy: 0.9907 - val_loss: 1.8255 - val_accuracy: 0.6256 -
val_top-5-accuracy: 0.9922
Epoch 40/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7485 - accuracy:
0.6915 - top-5-accuracy: 0.9920 - val_loss: 1.7775 - val_accuracy: 0.6500 -
val_top-5-accuracy: 0.9944
Epoch 41/100
64/64 [==============================] - 1s 19ms/step - loss: 1.7471 - accuracy:
```

```
0.6862 - top-5-accuracy: 0.9921 - val_loss: 1.7757 - val_accuracy: 0.6789 -
val_top-5-accuracy: 0.9956
Epoch 42/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7426 - accuracy:
0.6963 - top-5-accuracy: 0.9933 - val_loss: 1.7801 - val_accuracy: 0.6733 -
val_top-5-accuracy: 0.9944
Epoch 43/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7423 - accuracy:
0.7036 - top-5-accuracy: 0.9932 - val_loss: 1.7912 - val_accuracy: 0.6600 -
val_top-5-accuracy: 0.9922
Epoch 44/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7451 - accuracy:
0.6883 - top-5-accuracy: 0.9943 - val_loss: 1.7668 - val_accuracy: 0.6600 -
val_top-5-accuracy: 0.9967
Epoch 45/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7474 - accuracy:
0.6915 - top-5-accuracy: 0.9936 - val_loss: 1.8232 - val_accuracy: 0.6178 -
val_top-5-accuracy: 0.9967
Epoch 46/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7425 - accuracy:
0.6974 - top-5-accuracy: 0.9928 - val_loss: 1.7746 - val_accuracy: 0.6467 -
val_top-5-accuracy: 0.9933
Epoch 47/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7434 - accuracy:
0.6900 - top-5-accuracy: 0.9953 - val_loss: 1.7781 - val_accuracy: 0.6589 -
val_top-5-accuracy: 0.9911
Epoch 48/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7414 - accuracy:
0.6975 - top-5-accuracy: 0.9933 - val_loss: 1.7796 - val_accuracy: 0.6544 -
val_top-5-accuracy: 0.9944
Epoch 49/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7399 - accuracy:
0.7009 - top-5-accuracy: 0.9926 - val_loss: 1.7745 - val_accuracy: 0.6667 -
val_top-5-accuracy: 0.9922
Epoch 50/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7379 - accuracy:
0.7062 - top-5-accuracy: 0.9925 - val_loss: 1.8116 - val_accuracy: 0.6489 -
val_top-5-accuracy: 0.9978
Epoch 51/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7375 - accuracy:
0.7074 - top-5-accuracy: 0.9933 - val_loss: 1.7769 - val_accuracy: 0.6711 -
val_top-5-accuracy: 0.9900
Epoch 52/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7372 - accuracy:
0.7049 - top-5-accuracy: 0.9940 - val_loss: 1.8162 - val_accuracy: 0.6044 -
val_top-5-accuracy: 0.9889
Epoch 53/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7404 - accuracy:
```

```
0.7005 - top-5-accuracy: 0.9935 - val_loss: 1.7924 - val_accuracy: 0.6644 -
val_top-5-accuracy: 0.9956
Epoch 54/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7369 - accuracy:
0.7032 - top-5-accuracy: 0.9947 - val_loss: 1.7829 - val_accuracy: 0.6567 -
val_top-5-accuracy: 0.9956
Epoch 55/100
64/64 [==============================] - 1s 17ms/step - loss: 1.7365 - accuracy:
0.7080 - top-5-accuracy: 0.9944 - val_loss: 1.8059 - val_accuracy: 0.6533 -
val_top-5-accuracy: 0.9933
Epoch 56/100
64/64 [==============================] - 1s 18ms/step - loss: 1.7332 - accuracy:
0.7141 - top-5-accuracy: 0.9930 - val_loss: 1.7757 - val_accuracy: 0.6556 -
val_top-5-accuracy: 0.9967
32/32 [==============================] - 0s 5ms/step - loss: 1.7551 - accuracy:
0.7130 - top-5-accuracy: 0.9950
Test accuracy: 71.3%
```

[54]:
```
cleaning_network = create_label_cleaner_clipped(resnet44_classifier)
cleaning_network._name = "cleaning_network_resnet44_clipped"
history_cleaning_resnet44 = run_cleaning_label(cleaning_network)
```

```
Epoch 1/100
64/64 [==============================] - 11s 86ms/step - loss: 2.0538 -
accuracy: 0.3533 - top-5-accuracy: 0.8611 - val_loss: 1.9014 - val_accuracy:
0.5544 - val_top-5-accuracy: 0.9444
Epoch 2/100
64/64 [==============================] - 4s 69ms/step - loss: 1.9004 - accuracy:
0.5262 - top-5-accuracy: 0.9340 - val_loss: 1.8634 - val_accuracy: 0.5656 -
val_top-5-accuracy: 0.9511
Epoch 3/100
64/64 [==============================] - 5s 72ms/step - loss: 1.8496 - accuracy:
0.5944 - top-5-accuracy: 0.9481 - val_loss: 1.8116 - val_accuracy: 0.6433 -
val_top-5-accuracy: 0.9633
Epoch 4/100
64/64 [==============================] - 4s 65ms/step - loss: 1.8143 - accuracy:
0.6274 - top-5-accuracy: 0.9627 - val_loss: 1.8050 - val_accuracy: 0.6233 -
val_top-5-accuracy: 0.9633
Epoch 5/100
64/64 [==============================] - 4s 70ms/step - loss: 1.7960 - accuracy:
0.6573 - top-5-accuracy: 0.9679 - val_loss: 1.7781 - val_accuracy: 0.6800 -
val_top-5-accuracy: 0.9778
Epoch 6/100
64/64 [==============================] - 4s 66ms/step - loss: 1.7797 - accuracy:
0.6796 - top-5-accuracy: 0.9765 - val_loss: 1.7921 - val_accuracy: 0.6733 -
val_top-5-accuracy: 0.9744
Epoch 7/100
64/64 [==============================] - 4s 70ms/step - loss: 1.7684 - accuracy:
```

```
0.6906 - top-5-accuracy: 0.9806 - val_loss: 1.7684 - val_accuracy: 0.7000 -
val_top-5-accuracy: 0.9967
Epoch 8/100
64/64 [==============================] - 4s 66ms/step - loss: 1.7595 - accuracy:
0.7051 - top-5-accuracy: 0.9822 - val_loss: 1.7549 - val_accuracy: 0.6956 -
val_top-5-accuracy: 0.9900
Epoch 9/100
64/64 [==============================] - 4s 66ms/step - loss: 1.7504 - accuracy:
0.7199 - top-5-accuracy: 0.9858 - val_loss: 1.7613 - val_accuracy: 0.6856 -
val_top-5-accuracy: 0.9956
Epoch 10/100
64/64 [==============================] - 4s 66ms/step - loss: 1.7435 - accuracy:
0.7321 - top-5-accuracy: 0.9899 - val_loss: 1.7720 - val_accuracy: 0.6956 -
val_top-5-accuracy: 0.9900
Epoch 11/100
64/64 [==============================] - 4s 65ms/step - loss: 1.7392 - accuracy:
0.7312 - top-5-accuracy: 0.9889 - val_loss: 1.7693 - val_accuracy: 0.6933 -
val_top-5-accuracy: 0.9956
Epoch 12/100
64/64 [==============================] - 4s 65ms/step - loss: 1.7346 - accuracy:
0.7417 - top-5-accuracy: 0.9899 - val_loss: 1.7739 - val_accuracy: 0.6878 -
val_top-5-accuracy: 0.9900
Epoch 13/100
64/64 [==============================] - 4s 70ms/step - loss: 1.7317 - accuracy:
0.7457 - top-5-accuracy: 0.9898 - val_loss: 1.7611 - val_accuracy: 0.7156 -
val_top-5-accuracy: 0.9933
Epoch 14/100
64/64 [==============================] - 4s 65ms/step - loss: 1.7243 - accuracy:
0.7611 - top-5-accuracy: 0.9921 - val_loss: 1.7882 - val_accuracy: 0.6811 -
val_top-5-accuracy: 0.9967
Epoch 15/100
64/64 [==============================] - 4s 65ms/step - loss: 1.7156 - accuracy:
0.7756 - top-5-accuracy: 0.9933 - val_loss: 1.7840 - val_accuracy: 0.6844 -
val_top-5-accuracy: 0.9889
Epoch 16/100
64/64 [==============================] - 4s 65ms/step - loss: 1.7189 - accuracy:
0.7648 - top-5-accuracy: 0.9917 - val_loss: 1.7883 - val_accuracy: 0.6989 -
val_top-5-accuracy: 0.9967
Epoch 17/100
64/64 [==============================] - 5s 72ms/step - loss: 1.7118 - accuracy:
0.7805 - top-5-accuracy: 0.9937 - val_loss: 1.7295 - val_accuracy: 0.7433 -
val_top-5-accuracy: 0.9967
Epoch 18/100
64/64 [==============================] - 4s 65ms/step - loss: 1.7058 - accuracy:
0.7891 - top-5-accuracy: 0.9944 - val_loss: 1.7723 - val_accuracy: 0.6900 -
val_top-5-accuracy: 0.9911
Epoch 19/100
64/64 [==============================] - 4s 65ms/step - loss: 1.7068 - accuracy:
```

```
0.7872 - top-5-accuracy: 0.9956 - val_loss: 1.7464 - val_accuracy: 0.7344 -
val_top-5-accuracy: 0.9956
Epoch 20/100
64/64 [==============================] - 4s 65ms/step - loss: 1.7026 - accuracy:
0.7849 - top-5-accuracy: 0.9952 - val_loss: 1.7508 - val_accuracy: 0.7011 -
val_top-5-accuracy: 0.9956
Epoch 21/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6982 - accuracy:
0.7920 - top-5-accuracy: 0.9952 - val_loss: 1.7583 - val_accuracy: 0.7100 -
val_top-5-accuracy: 0.9956
Epoch 22/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6997 - accuracy:
0.7925 - top-5-accuracy: 0.9954 - val_loss: 1.7691 - val_accuracy: 0.7067 -
val_top-5-accuracy: 0.9922
Epoch 23/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6941 - accuracy:
0.8000 - top-5-accuracy: 0.9963 - val_loss: 1.7333 - val_accuracy: 0.7422 -
val_top-5-accuracy: 0.9989
Epoch 24/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6994 - accuracy:
0.7919 - top-5-accuracy: 0.9951 - val_loss: 1.7954 - val_accuracy: 0.6711 -
val_top-5-accuracy: 0.9956
Epoch 25/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6920 - accuracy:
0.7965 - top-5-accuracy: 0.9965 - val_loss: 1.7611 - val_accuracy: 0.7144 -
val_top-5-accuracy: 0.9978
Epoch 26/100
64/64 [==============================] - 4s 70ms/step - loss: 1.6842 - accuracy:
0.8115 - top-5-accuracy: 0.9968 - val_loss: 1.7343 - val_accuracy: 0.7556 -
val_top-5-accuracy: 0.9978
Epoch 27/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6837 - accuracy:
0.8116 - top-5-accuracy: 0.9975 - val_loss: 1.7969 - val_accuracy: 0.6844 -
val_top-5-accuracy: 0.9989
Epoch 28/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6960 - accuracy:
0.7942 - top-5-accuracy: 0.9968 - val_loss: 1.7725 - val_accuracy: 0.7144 -
val_top-5-accuracy: 0.9967
Epoch 29/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6838 - accuracy:
0.8101 - top-5-accuracy: 0.9960 - val_loss: 1.8172 - val_accuracy: 0.6400 -
val_top-5-accuracy: 0.9978
Epoch 30/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6768 - accuracy:
0.8231 - top-5-accuracy: 0.9972 - val_loss: 1.7499 - val_accuracy: 0.7378 -
val_top-5-accuracy: 0.9989
Epoch 31/100
64/64 [==============================] - 4s 67ms/step - loss: 1.6734 - accuracy:
```

```
0.8305 - top-5-accuracy: 0.9977 - val_loss: 1.7311 - val_accuracy: 0.7533 -
val_top-5-accuracy: 1.0000
Epoch 32/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6745 - accuracy:
0.8212 - top-5-accuracy: 0.9973 - val_loss: 1.7691 - val_accuracy: 0.7067 -
val_top-5-accuracy: 0.9967
Epoch 33/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6756 - accuracy:
0.8169 - top-5-accuracy: 0.9973 - val_loss: 1.7490 - val_accuracy: 0.7211 -
val_top-5-accuracy: 0.9967
Epoch 34/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6703 - accuracy:
0.8284 - top-5-accuracy: 0.9985 - val_loss: 1.7728 - val_accuracy: 0.7000 -
val_top-5-accuracy: 0.9978
Epoch 35/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6667 - accuracy:
0.8319 - top-5-accuracy: 0.9980 - val_loss: 1.8074 - val_accuracy: 0.6744 -
val_top-5-accuracy: 0.9967
Epoch 36/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6739 - accuracy:
0.8184 - top-5-accuracy: 0.9983 - val_loss: 1.7694 - val_accuracy: 0.7044 -
val_top-5-accuracy: 0.9978
Epoch 37/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6683 - accuracy:
0.8351 - top-5-accuracy: 0.9984 - val_loss: 1.7563 - val_accuracy: 0.7267 -
val_top-5-accuracy: 1.0000
Epoch 38/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6674 - accuracy:
0.8311 - top-5-accuracy: 0.9983 - val_loss: 1.7301 - val_accuracy: 0.7533 -
val_top-5-accuracy: 1.0000
Epoch 39/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6673 - accuracy:
0.8344 - top-5-accuracy: 0.9977 - val_loss: 1.7311 - val_accuracy: 0.7500 -
val_top-5-accuracy: 0.9989
Epoch 40/100
64/64 [==============================] - 4s 70ms/step - loss: 1.6639 - accuracy:
0.8389 - top-5-accuracy: 0.9981 - val_loss: 1.7226 - val_accuracy: 0.7678 -
val_top-5-accuracy: 0.9978
Epoch 41/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6608 - accuracy:
0.8417 - top-5-accuracy: 0.9988 - val_loss: 1.7596 - val_accuracy: 0.7222 -
val_top-5-accuracy: 0.9978
Epoch 42/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6627 - accuracy:
0.8432 - top-5-accuracy: 0.9985 - val_loss: 1.7420 - val_accuracy: 0.7256 -
val_top-5-accuracy: 0.9956
Epoch 43/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6571 - accuracy:
```

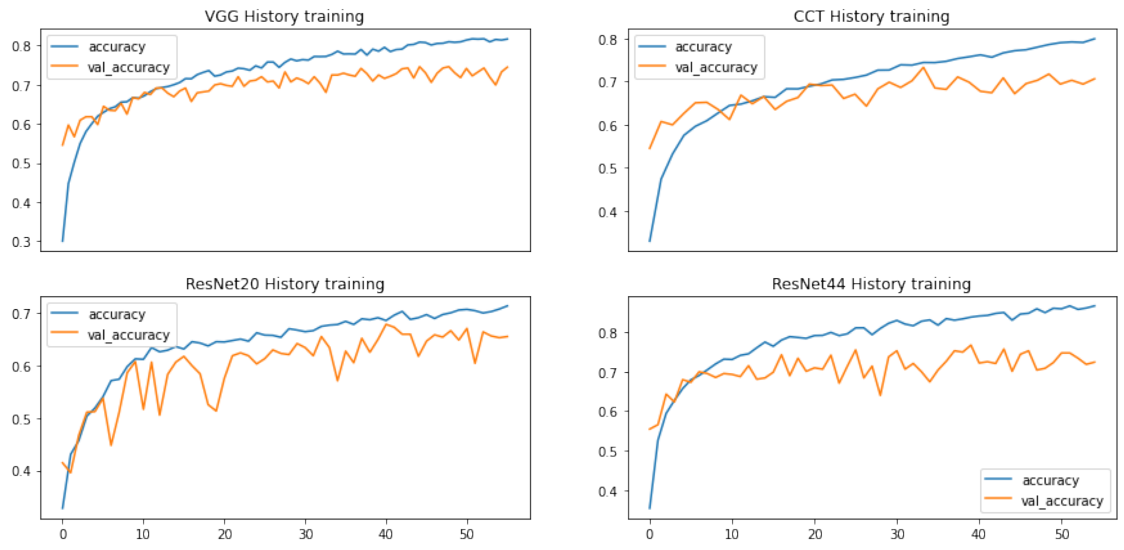
```
0.8483 - top-5-accuracy: 0.9980 - val_loss: 1.7520 - val_accuracy: 0.7211 -
val_top-5-accuracy: 0.9967
Epoch 44/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6554 - accuracy:
0.8506 - top-5-accuracy: 0.9990 - val_loss: 1.7260 - val_accuracy: 0.7578 -
val_top-5-accuracy: 1.0000
Epoch 45/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6663 - accuracy:
0.8311 - top-5-accuracy: 0.9983 - val_loss: 1.7696 - val_accuracy: 0.7011 -
val_top-5-accuracy: 1.0000
Epoch 46/100
64/64 [==============================] - 4s 67ms/step - loss: 1.6566 - accuracy:
0.8467 - top-5-accuracy: 0.9978 - val_loss: 1.7414 - val_accuracy: 0.7444 -
val_top-5-accuracy: 0.9989
Epoch 47/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6562 - accuracy:
0.8486 - top-5-accuracy: 0.9988 - val_loss: 1.7354 - val_accuracy: 0.7533 -
val_top-5-accuracy: 0.9978
Epoch 48/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6516 - accuracy:
0.8600 - top-5-accuracy: 0.9985 - val_loss: 1.7714 - val_accuracy: 0.7044 -
val_top-5-accuracy: 0.9967
Epoch 49/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6513 - accuracy:
0.8502 - top-5-accuracy: 0.9990 - val_loss: 1.7593 - val_accuracy: 0.7089 -
val_top-5-accuracy: 1.0000
Epoch 50/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6494 - accuracy:
0.8614 - top-5-accuracy: 0.9986 - val_loss: 1.7665 - val_accuracy: 0.7233 -
val_top-5-accuracy: 1.0000
Epoch 51/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6473 - accuracy:
0.8600 - top-5-accuracy: 0.9986 - val_loss: 1.7296 - val_accuracy: 0.7478 -
val_top-5-accuracy: 0.9978
Epoch 52/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6470 - accuracy:
0.8674 - top-5-accuracy: 0.9988 - val_loss: 1.7369 - val_accuracy: 0.7478 -
val_top-5-accuracy: 1.0000
Epoch 53/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6510 - accuracy:
0.8585 - top-5-accuracy: 0.9986 - val_loss: 1.7405 - val_accuracy: 0.7344 -
val_top-5-accuracy: 0.9978
Epoch 54/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6482 - accuracy:
0.8620 - top-5-accuracy: 0.9990 - val_loss: 1.7481 - val_accuracy: 0.7189 -
val_top-5-accuracy: 0.9967
Epoch 55/100
64/64 [==============================] - 4s 65ms/step - loss: 1.6467 - accuracy:
```

```
0.8674 - top-5-accuracy: 0.9985 - val_loss: 1.7580 - val_accuracy: 0.7244 -
val_top-5-accuracy: 1.0000
32/32 [==============================] - 0s 10ms/step - loss: 1.7155 - accuracy:
0.7720 - top-5-accuracy: 0.9950
Test accuracy: 77.2%
```

Training history of our cleaning label models :

```
[91]: plt.rcParams["figure.figsize"] = (15,7)
fig, axs = plt.subplots(2,2)
pd.DataFrame(history_cleaning_vgg.history)[["accuracy","val_accuracy"]].plot(ax␣
 ↪= axs[0,0])
axs[0,0].set_title("VGG History training")
axs[0,0].get_xaxis().set_ticks([])
pd.DataFrame(history_cleaning_cct.history)[["accuracy","val_accuracy"]].
 ↪plot(ax=axs[0,1])
axs[0,1].set_title("CCT History training")
axs[0,1].get_xaxis().set_ticks([])
pd.DataFrame(history_cleaning_resnet20.history)[["accuracy","val_accuracy"]].
 ↪plot(ax=axs[1,0])
axs[1,0].set_title("ResNet20 History training")
pd.DataFrame(history_cleaning_resnet44.history)[["accuracy","val_accuracy"]].
 ↪plot(ax = axs[1,1])
axs[1,1].set_title("ResNet44 History training")
```

```
[91]: Text(0.5, 1.0, 'ResNet44 History training')
```



Accruacy comparaison between the model on the test set

Model	Accuracy	#Params	Time per epoch
VGG	76.6%	0.82M	1s
CCT	70.5%	0.65M	2s
ResNet20	71.3%	0.12M	1s
ResNet44	77.2%	0.71M	4s

Again VGG as a very good accuracy and very low computational time.

Train with the new predicted label

Once we have trained our label cleaning network, we feed the noisy label to get the new corrected label that will be use for training. Again we separate our dataset into a training set (composed of the 45,000 last images and a mix of corrected labels and cleaned labels) and a testing set (composed of the 5,000 first images and cleaned labels). Since we are fine tuning, our Model I, we reduce the learning rate to $1e-4$ to end up with a better accuracy.

```
[24]: learning_rate=1e-3
      def run_experiment2(model,name):
          # Compute the new predicted label from the base Encoder used
          cleaning_network = create_label_cleaner_clipped(model)
          cleaning_network.load_weights("./tmp/
       ↪checkpoint_cleaning_network_"+name+"_clipped")
          train = imgs_prep[5000:50000]
          test = imgs_prep[:5000]
          train_labels = smooth_labels(keras.utils.to_categorical(labels[5000:50000]))
          test_cleaned_labels = smooth_labels(keras.utils.to_categorical(labels[:
       ↪5000]))
          train_cleaned_labels = tf.argmax(cleaning_network.predict([train[2000:
       ↪],train_labels[2000:]]),axis=1)
          train_cleaned_labels = smooth_labels(keras.utils.to_categorical(tf.
       ↪concat([labels[5000:7000],train_cleaned_labels],axis=0)))
          optimizer = tfa.optimizers.AdamW(
              learning_rate=learning_rate, weight_decay=weight_decay
          )
          model.compile(
              optimizer=optimizer,
              loss=keras.losses.CategoricalCrossentropy(from_logits=True),
              metrics=[
                  keras.metrics.CategoricalAccuracy(name="accuracy"),
                  keras.metrics.TopKCategoricalAccuracy(5, name="top-5-accuracy"),
              ],
          )

          checkpoint_filepath = "./tmp/checkpoint_"+str(model.name)
          checkpoint_callback = keras.callbacks.ModelCheckpoint(
              checkpoint_filepath,
              monitor="val_accuracy",
              save_best_only=True,
              save_weights_only=True,
          )
          early_stoping_callback = keras.callbacks.
      ↪EarlyStopping(monitor='val_accuracy', patience=10)

          history = model.fit(
              x=train,
              y=train_cleaned_labels,
              batch_size=batch_size,
              epochs=num_epochs,
              validation_split=0.1,
              callbacks=[checkpoint_callback,early_stoping_callback],
          )

          model.load_weights(checkpoint_filepath)
          _, accuracy, top_5_accuracy = model.evaluate(test, test_cleaned_labels)
          print(f"Test accuracy: {round(accuracy * 100, 2)}%")
          print(f"Test top 5 accuracy: {round(top_5_accuracy * 100, 2)}%")

          return history
```

[93]:
```
learning_rate = 1e-4
vgg_classifier = create_vgg_classifier(3)
vgg_classifier.load_weights("./tmp/checkpoint_model_vgg_classifier")
vgg_classifier._name = 'model_vgg_classifier_2'
history_vgg_2 = run_experiment2(vgg_classifier,"vgg")
```

```
Epoch 1/100
317/317 [==============================] - 6s 17ms/step - loss: 1.1295 -
accuracy: 0.7285 - top-5-accuracy: 0.9776 - val_loss: 0.9225 - val_accuracy:
0.8267 - val_top-5-accuracy: 0.9911
Epoch 2/100
317/317 [==============================] - 5s 15ms/step - loss: 1.0513 -
accuracy: 0.7607 - top-5-accuracy: 0.9838 - val_loss: 0.9540 - val_accuracy:
0.7987 - val_top-5-accuracy: 0.9893
Epoch 3/100
317/317 [==============================] - 5s 15ms/step - loss: 1.0201 -
accuracy: 0.7738 - top-5-accuracy: 0.9856 - val_loss: 0.9721 - val_accuracy:
0.7911 - val_top-5-accuracy: 0.9891
Epoch 4/100
317/317 [==============================] - 5s 15ms/step - loss: 1.0068 -
accuracy: 0.7760 - top-5-accuracy: 0.9873 - val_loss: 0.9593 - val_accuracy:
0.7953 - val_top-5-accuracy: 0.9891
Epoch 5/100
317/317 [==============================] - 5s 15ms/step - loss: 0.9884 -
accuracy: 0.7864 - top-5-accuracy: 0.9875 - val_loss: 0.9677 - val_accuracy:
0.7902 - val_top-5-accuracy: 0.9898
```

```
Epoch 6/100
317/317 [==============================] - 5s 15ms/step - loss: 0.9846 -
accuracy: 0.7871 - top-5-accuracy: 0.9883 - val_loss: 0.9205 - val_accuracy:
0.8122 - val_top-5-accuracy: 0.9907
Epoch 7/100
317/317 [==============================] - 5s 15ms/step - loss: 0.9737 -
accuracy: 0.7924 - top-5-accuracy: 0.9884 - val_loss: 0.9625 - val_accuracy:
0.7893 - val_top-5-accuracy: 0.9884
Epoch 8/100
317/317 [==============================] - 5s 15ms/step - loss: 0.9675 -
accuracy: 0.7923 - top-5-accuracy: 0.9896 - val_loss: 0.9025 - val_accuracy:
0.8213 - val_top-5-accuracy: 0.9913
Epoch 9/100
317/317 [==============================] - 5s 15ms/step - loss: 0.9659 -
accuracy: 0.7937 - top-5-accuracy: 0.9886 - val_loss: 0.9427 - val_accuracy:
0.8040 - val_top-5-accuracy: 0.9893
Epoch 10/100
317/317 [==============================] - 5s 15ms/step - loss: 0.9553 -
accuracy: 0.7995 - top-5-accuracy: 0.9892 - val_loss: 0.9362 - val_accuracy:
0.8082 - val_top-5-accuracy: 0.9878
Epoch 11/100
317/317 [==============================] - 5s 15ms/step - loss: 0.9530 -
accuracy: 0.8019 - top-5-accuracy: 0.9887 - val_loss: 0.8938 - val_accuracy:
0.8204 - val_top-5-accuracy: 0.9907
157/157 [==============================] - 1s 4ms/step - loss: 1.0234 -
accuracy: 0.7798 - top-5-accuracy: 0.9832
Test accuracy: 77.98%
Test top 5 accuracy: 98.32%
```

[94]:
```
cct_classifier = create_cct_classifier()
cct_classifier.load_weights("./tmp/checkpoint_model_cct_classifier")
cct_classifier._name = 'model_cct_classifier_2'
history_cct_2 = run_experiment2(cct_classifier,"cct")
```

```
Epoch 1/100
317/317 [==============================] - 13s 30ms/step - loss: 1.1015 -
accuracy: 0.7311 - top-5-accuracy: 0.9827 - val_loss: 0.9502 - val_accuracy:
0.7962 - val_top-5-accuracy: 0.9913
Epoch 2/100
317/317 [==============================] - 9s 27ms/step - loss: 1.0371 -
accuracy: 0.7543 - top-5-accuracy: 0.9878 - val_loss: 0.9880 - val_accuracy:
0.7764 - val_top-5-accuracy: 0.9920
Epoch 3/100
317/317 [==============================] - 9s 28ms/step - loss: 1.0153 -
accuracy: 0.7658 - top-5-accuracy: 0.9893 - val_loss: 0.9320 - val_accuracy:
0.7998 - val_top-5-accuracy: 0.9940
Epoch 4/100
317/317 [==============================] - 9s 28ms/step - loss: 0.9955 -
```

```
accuracy: 0.7740 - top-5-accuracy: 0.9897 - val_loss: 0.9076 - val_accuracy:
0.8133 - val_top-5-accuracy: 0.9956
Epoch 5/100
317/317 [==============================] - 9s 27ms/step - loss: 0.9885 -
accuracy: 0.7762 - top-5-accuracy: 0.9905 - val_loss: 0.9257 - val_accuracy:
0.8011 - val_top-5-accuracy: 0.9938
Epoch 6/100
317/317 [==============================] - 9s 28ms/step - loss: 0.9741 -
accuracy: 0.7843 - top-5-accuracy: 0.9905 - val_loss: 0.9610 - val_accuracy:
0.7876 - val_top-5-accuracy: 0.9944
Epoch 7/100
317/317 [==============================] - 9s 28ms/step - loss: 0.9680 -
accuracy: 0.7877 - top-5-accuracy: 0.9908 - val_loss: 0.9066 - val_accuracy:
0.8120 - val_top-5-accuracy: 0.9933
Epoch 8/100
317/317 [==============================] - 9s 27ms/step - loss: 0.9684 -
accuracy: 0.7861 - top-5-accuracy: 0.9910 - val_loss: 0.9320 - val_accuracy:
0.7993 - val_top-5-accuracy: 0.9927
Epoch 9/100
317/317 [==============================] - 9s 27ms/step - loss: 0.9633 -
accuracy: 0.7856 - top-5-accuracy: 0.9918 - val_loss: 0.9390 - val_accuracy:
0.7924 - val_top-5-accuracy: 0.9933
Epoch 10/100
317/317 [==============================] - 9s 27ms/step - loss: 0.9553 -
accuracy: 0.7919 - top-5-accuracy: 0.9923 - val_loss: 0.9330 - val_accuracy:
0.7960 - val_top-5-accuracy: 0.9942
Epoch 11/100
317/317 [==============================] - 9s 28ms/step - loss: 0.9523 -
accuracy: 0.7928 - top-5-accuracy: 0.9916 - val_loss: 0.9009 - val_accuracy:
0.8127 - val_top-5-accuracy: 0.9962
Epoch 12/100
317/317 [==============================] - 9s 27ms/step - loss: 0.9470 -
accuracy: 0.7968 - top-5-accuracy: 0.9919 - val_loss: 0.9360 - val_accuracy:
0.7967 - val_top-5-accuracy: 0.9927
Epoch 13/100
317/317 [==============================] - 9s 27ms/step - loss: 0.9500 -
accuracy: 0.7953 - top-5-accuracy: 0.9918 - val_loss: 0.9208 - val_accuracy:
0.8049 - val_top-5-accuracy: 0.9951
Epoch 14/100
317/317 [==============================] - 9s 28ms/step - loss: 0.9379 -
accuracy: 0.7998 - top-5-accuracy: 0.9920 - val_loss: 0.9290 - val_accuracy:
0.8013 - val_top-5-accuracy: 0.9940
157/157 [==============================] - 1s 7ms/step - loss: 1.1969 -
accuracy: 0.7024 - top-5-accuracy: 0.9764
Test accuracy: 70.24%
Test top 5 accuracy: 97.64%
```

[95]:
```
resnet20_classifier = create_resnet_classifier(1)
resnet20_classifier.load_weights("./tmp/checkpoint_model_resnet20_classifier")
resnet20_classifier._name = 'model_resnet20_classifier_2'
history_resnet20_2 = run_experiment2(resnet20_classifier,"resnet20")
```

```
Epoch 1/100
317/317 [==============================] - 8s 19ms/step - loss: 1.1995 -
accuracy: 0.7056 - top-5-accuracy: 0.9776 - val_loss: 1.2329 - val_accuracy:
0.6962 - val_top-5-accuracy: 0.9687
Epoch 2/100
317/317 [==============================] - 5s 17ms/step - loss: 1.1361 -
accuracy: 0.7255 - top-5-accuracy: 0.9810 - val_loss: 1.3574 - val_accuracy:
0.6400 - val_top-5-accuracy: 0.9647
Epoch 3/100
317/317 [==============================] - 5s 17ms/step - loss: 1.1176 -
accuracy: 0.7300 - top-5-accuracy: 0.9820 - val_loss: 1.1834 - val_accuracy:
0.6873 - val_top-5-accuracy: 0.9853
Epoch 4/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0984 -
accuracy: 0.7402 - top-5-accuracy: 0.9840 - val_loss: 1.1321 - val_accuracy:
0.7078 - val_top-5-accuracy: 0.9716
Epoch 5/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0913 -
accuracy: 0.7418 - top-5-accuracy: 0.9829 - val_loss: 1.1184 - val_accuracy:
0.7233 - val_top-5-accuracy: 0.9896
Epoch 6/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0877 -
accuracy: 0.7430 - top-5-accuracy: 0.9847 - val_loss: 1.0251 - val_accuracy:
0.7622 - val_top-5-accuracy: 0.9860
Epoch 7/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0778 -
accuracy: 0.7460 - top-5-accuracy: 0.9844 - val_loss: 1.1713 - val_accuracy:
0.6856 - val_top-5-accuracy: 0.9780
Epoch 8/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0760 -
accuracy: 0.7473 - top-5-accuracy: 0.9851 - val_loss: 0.9973 - val_accuracy:
0.7658 - val_top-5-accuracy: 0.9913
Epoch 9/100
317/317 [==============================] - 6s 18ms/step - loss: 1.0733 -
accuracy: 0.7470 - top-5-accuracy: 0.9848 - val_loss: 1.1082 - val_accuracy:
0.7229 - val_top-5-accuracy: 0.9867
Epoch 10/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0674 -
accuracy: 0.7498 - top-5-accuracy: 0.9842 - val_loss: 1.4099 - val_accuracy:
0.6271 - val_top-5-accuracy: 0.9749
Epoch 11/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0663 -
accuracy: 0.7506 - top-5-accuracy: 0.9841 - val_loss: 1.0526 - val_accuracy:
```

```
0.7438 - val_top-5-accuracy: 0.9893
Epoch 12/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0605 -
accuracy: 0.7517 - top-5-accuracy: 0.9844 - val_loss: 1.1255 - val_accuracy:
0.7189 - val_top-5-accuracy: 0.9811
Epoch 13/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0614 -
accuracy: 0.7534 - top-5-accuracy: 0.9846 - val_loss: 1.0978 - val_accuracy:
0.7280 - val_top-5-accuracy: 0.9867
Epoch 14/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0588 -
accuracy: 0.7523 - top-5-accuracy: 0.9851 - val_loss: 1.1349 - val_accuracy:
0.7133 - val_top-5-accuracy: 0.9818
Epoch 15/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0540 -
accuracy: 0.7547 - top-5-accuracy: 0.9850 - val_loss: 1.2822 - val_accuracy:
0.6484 - val_top-5-accuracy: 0.9824
Epoch 16/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0538 -
accuracy: 0.7532 - top-5-accuracy: 0.9855 - val_loss: 1.0160 - val_accuracy:
0.7627 - val_top-5-accuracy: 0.9878
Epoch 17/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0575 -
accuracy: 0.7547 - top-5-accuracy: 0.9855 - val_loss: 1.1205 - val_accuracy:
0.7162 - val_top-5-accuracy: 0.9867
Epoch 18/100
317/317 [==============================] - 5s 17ms/step - loss: 1.0504 -
accuracy: 0.7555 - top-5-accuracy: 0.9865 - val_loss: 1.1202 - val_accuracy:
0.7224 - val_top-5-accuracy: 0.9829
157/157 [==============================] - 1s 4ms/step - loss: 1.3594 -
accuracy: 0.6344 - top-5-accuracy: 0.9576
Test accuracy: 63.44%
Test top 5 accuracy: 95.76%
```

[97]:
```python
resnet44_classifier = create_resnet_classifier(7)
resnet44_classifier.load_weights("./tmp/checkpoint_model_resnet44_classifier")
resnet44_classifier._name='model_resnet44_classifier_2'
history_resnet44_2 = run_experiment2(resnet44_classifier,"resnet44")
```

```
Epoch 1/100
317/317 [==============================] - 28s 72ms/step - loss: 1.0967 -
accuracy: 0.7580 - top-5-accuracy: 0.9766 - val_loss: 1.0902 - val_accuracy:
0.7416 - val_top-5-accuracy: 0.9818
Epoch 2/100
317/317 [==============================] - 22s 68ms/step - loss: 1.0226 -
accuracy: 0.7837 - top-5-accuracy: 0.9831 - val_loss: 0.9781 - val_accuracy:
0.7936 - val_top-5-accuracy: 0.9862
Epoch 3/100
```

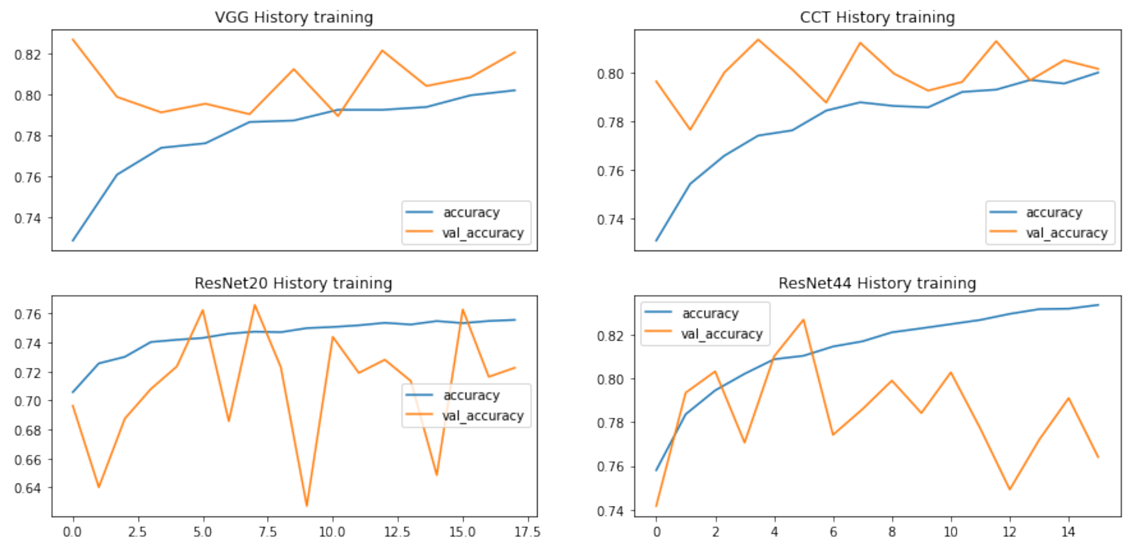
```
317/317 [==============================] - 21s 68ms/step - loss: 0.9949 -
accuracy: 0.7946 - top-5-accuracy: 0.9842 - val_loss: 0.9491 - val_accuracy:
0.8033 - val_top-5-accuracy: 0.9876
Epoch 4/100
317/317 [==============================] - 21s 67ms/step - loss: 0.9761 -
accuracy: 0.8022 - top-5-accuracy: 0.9851 - val_loss: 1.0334 - val_accuracy:
0.7707 - val_top-5-accuracy: 0.9842
Epoch 5/100
317/317 [==============================] - 22s 68ms/step - loss: 0.9595 -
accuracy: 0.8089 - top-5-accuracy: 0.9865 - val_loss: 0.9205 - val_accuracy:
0.8102 - val_top-5-accuracy: 0.9891
Epoch 6/100
317/317 [==============================] - 22s 68ms/step - loss: 0.9540 -
accuracy: 0.8105 - top-5-accuracy: 0.9859 - val_loss: 0.9058 - val_accuracy:
0.8271 - val_top-5-accuracy: 0.9896
Epoch 7/100
317/317 [==============================] - 21s 67ms/step - loss: 0.9445 -
accuracy: 0.8148 - top-5-accuracy: 0.9871 - val_loss: 1.0103 - val_accuracy:
0.7742 - val_top-5-accuracy: 0.9831
Epoch 8/100
317/317 [==============================] - 21s 67ms/step - loss: 0.9390 -
accuracy: 0.8171 - top-5-accuracy: 0.9869 - val_loss: 0.9938 - val_accuracy:
0.7860 - val_top-5-accuracy: 0.9833
Epoch 9/100
317/317 [==============================] - 21s 67ms/step - loss: 0.9342 -
accuracy: 0.8213 - top-5-accuracy: 0.9864 - val_loss: 0.9615 - val_accuracy:
0.7991 - val_top-5-accuracy: 0.9849
Epoch 10/100
317/317 [==============================] - 21s 67ms/step - loss: 0.9264 -
accuracy: 0.8231 - top-5-accuracy: 0.9873 - val_loss: 0.9900 - val_accuracy:
0.7842 - val_top-5-accuracy: 0.9851
Epoch 11/100
317/317 [==============================] - 21s 67ms/step - loss: 0.9212 -
accuracy: 0.8250 - top-5-accuracy: 0.9880 - val_loss: 0.9471 - val_accuracy:
0.8029 - val_top-5-accuracy: 0.9880
Epoch 12/100
317/317 [==============================] - 21s 67ms/step - loss: 0.9162 -
accuracy: 0.8270 - top-5-accuracy: 0.9880 - val_loss: 1.0176 - val_accuracy:
0.7771 - val_top-5-accuracy: 0.9804
Epoch 13/100
317/317 [==============================] - 21s 67ms/step - loss: 0.9150 -
accuracy: 0.8298 - top-5-accuracy: 0.9888 - val_loss: 1.0756 - val_accuracy:
0.7491 - val_top-5-accuracy: 0.9736
Epoch 14/100
317/317 [==============================] - 21s 68ms/step - loss: 0.9053 -
accuracy: 0.8319 - top-5-accuracy: 0.9888 - val_loss: 1.0358 - val_accuracy:
0.7720 - val_top-5-accuracy: 0.9824
Epoch 15/100
```

```
317/317 [==============================] - 21s 67ms/step - loss: 0.9086 -
accuracy: 0.8321 - top-5-accuracy: 0.9885 - val_loss: 0.9818 - val_accuracy:
0.7911 - val_top-5-accuracy: 0.9820
Epoch 16/100
317/317 [==============================] - 21s 67ms/step - loss: 0.9014 -
accuracy: 0.8339 - top-5-accuracy: 0.9888 - val_loss: 1.0373 - val_accuracy:
0.7640 - val_top-5-accuracy: 0.9836
157/157 [==============================] - 1s 9ms/step - loss: 0.9460 -
accuracy: 0.8120 - top-5-accuracy: 0.9834
Test accuracy: 81.2%
Test top 5 accuracy: 98.34%
```

Training history of our cleaning label models :

```
[98]: plt.rcParams["figure.figsize"] = (15,7)
fig, axs = plt.subplots(2,2)
pd.DataFrame(history_vgg_2.history)[["accuracy","val_accuracy"]].plot(ax =␣
 ↪axs[0,0])
axs[0,0].set_title("VGG History training")
axs[0,0].get_xaxis().set_ticks([])
pd.DataFrame(history_cct_2.history)[["accuracy","val_accuracy"]].
 ↪plot(ax=axs[0,1])
axs[0,1].set_title("CCT History training")
axs[0,1].get_xaxis().set_ticks([])
pd.DataFrame(history_resnet20_2.history)[["accuracy","val_accuracy"]].
 ↪plot(ax=axs[1,0])
axs[1,0].set_title("ResNet20 History training")
pd.DataFrame(history_resnet44_2.history)[["accuracy","val_accuracy"]].plot(ax =␣
 ↪axs[1,1])
axs[1,1].set_title("ResNet44 History training")
```

```
[98]: Text(0.5, 1.0, 'ResNet44 History training')
```



Accruacy comparaison between the model on the test set

Model	Accuracy	#Params	Time per epoch
VGG	77.98%	0.8M	5s
CCT	70.24%	0.63M	9s
ResNet20	63.44%	0.12M	5s
ResNet44	81.2%	0.69M	21s

Again VGG as a very good accuracy and very low computational time.

Train with a mixup augmentation data (mixup: BEYOND EMPIRICAL RISK MINIMIZATION)

Implement the mixup data augmentation

[25]:
```
def sample_beta_distribution(size, concentration_0=1., concentration_1=1.):
    gamma_1_sample = tf.random.gamma(shape=[size], alpha=concentration_1)
    gamma_2_sample = tf.random.gamma(shape=[size], alpha=concentration_0)
    return gamma_1_sample / (gamma_1_sample + gamma_2_sample)


def mix_up(ds_one, ds_two, alpha=0.2):
    # Unpack two datasets
    images_one, labels_one = ds_one
    images_two, labels_two = ds_two
    batch_size = tf.shape(images_one)[0]

    # Sample lambda and reshape it to do the mixup
    l = sample_beta_distribution(batch_size, alpha, alpha)
    x_l = tf.reshape(l, (batch_size, 1, 1, 1))
    y_l = tf.reshape(l, (batch_size, 1))

    # Perform mixup on both images and labels by combining a pair of images/
 ↪labels
    # (one from each dataset) into one image/label
    images = images_one * x_l + images_two * (1 - x_l)
    labels = labels_one * y_l + labels_two * (1 - y_l)
    return (images, labels)
```

[26]:
```
def run_experiment3(model,name):
    cleaning_network = create_label_cleaner_clipped(model)
    cleaning_network.load_weights("./tmp/
 ↪checkpoint_cleaning_network_"+name+"_clipped")
    train = imgs_prep[5000:50000]
    test = imgs_prep[:5000]
    train_labels = smooth_labels(keras.utils.to_categorical(labels[5000:50000]))
    test_cleaned_labels = smooth_labels(keras.utils.to_categorical(labels[:
↪5000]))
    train_cleaned_labels = tf.argmax(cleaning_network.predict([train[2000:
↪],train_labels[2000:]]),axis=1)
    train_cleaned_labels = smooth_labels(keras.utils.to_categorical(tf.
↪concat([labels[5000:7000],train_cleaned_labels],axis=0)))
    optimizer = tfa.optimizers.AdamW(
        learning_rate=learning_rate, weight_decay=weight_decay
    )

    val_samples = 2000
    val, val_cleaned_labels = train[:val_samples], train_cleaned_labels[:
↪val_samples]
    new_train, new_train_cleaned_labels = train[val_samples:],␣
↪train_cleaned_labels[val_samples:]

    train_ds_one = (
        tf.data.Dataset.from_tensor_slices((new_train,␣
↪new_train_cleaned_labels))
        .shuffle(batch_size * 100)
        .batch(batch_size)
    )
    train_ds_two = (
        tf.data.Dataset.from_tensor_slices((new_train,␣
↪new_train_cleaned_labels))
        .shuffle(batch_size * 100)
        .batch(batch_size)
    )
    # Because we will be mixing up the images and their corresponding labels,␣
↪we will be
    # combining two shuffled datasets from the same training data.
    train_ds = tf.data.Dataset.zip((train_ds_one, train_ds_two))

    val_ds = tf.data.Dataset.from_tensor_slices((val, val_cleaned_labels)).
↪batch(batch_size)

    test_ds = tf.data.Dataset.from_tensor_slices((test, test_cleaned_labels)).
↪batch(batch_size)

    train_ds_mu = train_ds.map(
        lambda ds_one, ds_two: mix_up(ds_one, ds_two, alpha=0.2),␣
↪num_parallel_calls=tf.data.AUTOTUNE
    )
    optimizer = tfa.optimizers.AdamW(
        learning_rate=learning_rate, weight_decay=weight_decay
    )
    model.compile(
        optimizer=optimizer,
        loss=keras.losses.CategoricalCrossentropy(from_logits=True),
        metrics=[
            keras.metrics.CategoricalAccuracy(name="accuracy"),
            keras.metrics.TopKCategoricalAccuracy(5, name="top-5-accuracy"),
        ],
    )

    checkpoint_filepath = "./tmp/checkpoint_"+str(model.name)
    checkpoint_callback = keras.callbacks.ModelCheckpoint(
        checkpoint_filepath,
        monitor="accuracy",
        save_best_only=True,
        save_weights_only=True,
    )
    early_stoping_callback = keras.callbacks.EarlyStopping(monitor='accuracy',␣
↪patience=5)
    history = model.fit(
        train_ds_mu,
        validation_data=val_ds,
        batch_size=batch_size,
        epochs=num_epochs,
        callbacks=[checkpoint_callback,early_stoping_callback],
    )

    model.load_weights(checkpoint_filepath)
    _, accuracy, top_5_accuracy = model.evaluate(test_ds)
    print(f"Test accuracy: {round(accuracy * 100, 2)}%")
    print(f"Test top 5 accuracy: {round(top_5_accuracy * 100, 2)}%")

    return history
```

[79]:
```
learning_rate = 1e-4
vgg_classifier = create_vgg_classifier(3)
vgg_classifier._name = 'model_3_vgg_classifier'
vgg_classifier.load_weights("./tmp/checkpoint_model_vgg_classifier")
history = run_experiment3(vgg_classifier,"vgg")
```

```
Epoch 1/100
336/336 [==============================] - 7s 17ms/step - loss: 1.3476 -
accuracy: 0.6963 - top-5-accuracy: 0.9637 - val_loss: 1.0677 - val_accuracy:
0.7560 - val_top-5-accuracy: 0.9785
Epoch 2/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2886 -
accuracy: 0.7195 - top-5-accuracy: 0.9722 - val_loss: 1.1001 - val_accuracy:
0.7445 - val_top-5-accuracy: 0.9800
```

```
Epoch 3/100
336/336 [==============================] - 5s 16ms/step - loss: 1.2646 -
accuracy: 0.7295 - top-5-accuracy: 0.9754 - val_loss: 1.1320 - val_accuracy:
0.7245 - val_top-5-accuracy: 0.9740
Epoch 4/100
336/336 [==============================] - 5s 16ms/step - loss: 1.2519 -
accuracy: 0.7380 - top-5-accuracy: 0.9771 - val_loss: 1.0906 - val_accuracy:
0.7445 - val_top-5-accuracy: 0.9750
Epoch 5/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2429 -
accuracy: 0.7417 - top-5-accuracy: 0.9763 - val_loss: 1.0643 - val_accuracy:
0.7485 - val_top-5-accuracy: 0.9800
Epoch 6/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2378 -
accuracy: 0.7417 - top-5-accuracy: 0.9771 - val_loss: 1.0540 - val_accuracy:
0.7560 - val_top-5-accuracy: 0.9790
Epoch 7/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2325 -
accuracy: 0.7446 - top-5-accuracy: 0.9779 - val_loss: 1.0329 - val_accuracy:
0.7720 - val_top-5-accuracy: 0.9785
Epoch 8/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2218 -
accuracy: 0.7513 - top-5-accuracy: 0.9779 - val_loss: 1.0718 - val_accuracy:
0.7545 - val_top-5-accuracy: 0.9790
Epoch 9/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2199 -
accuracy: 0.7507 - top-5-accuracy: 0.9782 - val_loss: 1.0962 - val_accuracy:
0.7395 - val_top-5-accuracy: 0.9740
Epoch 10/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2132 -
accuracy: 0.7572 - top-5-accuracy: 0.9783 - val_loss: 1.0264 - val_accuracy:
0.7675 - val_top-5-accuracy: 0.9810
Epoch 11/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2081 -
accuracy: 0.7590 - top-5-accuracy: 0.9791 - val_loss: 1.0616 - val_accuracy:
0.7655 - val_top-5-accuracy: 0.9780
Epoch 12/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2070 -
accuracy: 0.7592 - top-5-accuracy: 0.9791 - val_loss: 1.1147 - val_accuracy:
0.7305 - val_top-5-accuracy: 0.9745
Epoch 13/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2055 -
accuracy: 0.7575 - top-5-accuracy: 0.9785 - val_loss: 1.1227 - val_accuracy:
0.7395 - val_top-5-accuracy: 0.9685
Epoch 14/100
336/336 [==============================] - 5s 16ms/step - loss: 1.2060 -
accuracy: 0.7612 - top-5-accuracy: 0.9796 - val_loss: 1.0720 - val_accuracy:
0.7580 - val_top-5-accuracy: 0.9775
```

```
Epoch 15/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2027 -
accuracy: 0.7618 - top-5-accuracy: 0.9798 - val_loss: 1.1322 - val_accuracy:
0.7380 - val_top-5-accuracy: 0.9720
Epoch 16/100
336/336 [==============================] - 5s 15ms/step - loss: 1.2012 -
accuracy: 0.7606 - top-5-accuracy: 0.9809 - val_loss: 1.1349 - val_accuracy:
0.7290 - val_top-5-accuracy: 0.9710
Epoch 17/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1977 -
accuracy: 0.7604 - top-5-accuracy: 0.9798 - val_loss: 1.0928 - val_accuracy:
0.7540 - val_top-5-accuracy: 0.9680
Epoch 18/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1848 -
accuracy: 0.7684 - top-5-accuracy: 0.9811 - val_loss: 1.1112 - val_accuracy:
0.7440 - val_top-5-accuracy: 0.9705
Epoch 19/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1882 -
accuracy: 0.7667 - top-5-accuracy: 0.9813 - val_loss: 1.0811 - val_accuracy:
0.7465 - val_top-5-accuracy: 0.9760
Epoch 20/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1921 -
accuracy: 0.7667 - top-5-accuracy: 0.9808 - val_loss: 1.0888 - val_accuracy:
0.7470 - val_top-5-accuracy: 0.9730
Epoch 21/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1857 -
accuracy: 0.7684 - top-5-accuracy: 0.9813 - val_loss: 1.0839 - val_accuracy:
0.7505 - val_top-5-accuracy: 0.9715
Epoch 22/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1850 -
accuracy: 0.7695 - top-5-accuracy: 0.9804 - val_loss: 1.0991 - val_accuracy:
0.7425 - val_top-5-accuracy: 0.9750
Epoch 23/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1854 -
accuracy: 0.7700 - top-5-accuracy: 0.9806 - val_loss: 1.1123 - val_accuracy:
0.7325 - val_top-5-accuracy: 0.9765
Epoch 24/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1794 -
accuracy: 0.7720 - top-5-accuracy: 0.9807 - val_loss: 1.0732 - val_accuracy:
0.7580 - val_top-5-accuracy: 0.9730
Epoch 25/100
336/336 [==============================] - 5s 16ms/step - loss: 1.1837 -
accuracy: 0.7701 - top-5-accuracy: 0.9806 - val_loss: 1.1068 - val_accuracy:
0.7490 - val_top-5-accuracy: 0.9730
Epoch 26/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1755 -
accuracy: 0.7728 - top-5-accuracy: 0.9816 - val_loss: 1.1086 - val_accuracy:
0.7375 - val_top-5-accuracy: 0.9775
```

```
Epoch 27/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1820 -
accuracy: 0.7729 - top-5-accuracy: 0.9807 - val_loss: 1.0677 - val_accuracy:
0.7565 - val_top-5-accuracy: 0.9780
Epoch 28/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1801 -
accuracy: 0.7724 - top-5-accuracy: 0.9812 - val_loss: 1.0858 - val_accuracy:
0.7630 - val_top-5-accuracy: 0.9785
Epoch 29/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1820 -
accuracy: 0.7711 - top-5-accuracy: 0.9814 - val_loss: 1.0487 - val_accuracy:
0.7705 - val_top-5-accuracy: 0.9815
Epoch 30/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1798 -
accuracy: 0.7729 - top-5-accuracy: 0.9804 - val_loss: 1.1264 - val_accuracy:
0.7360 - val_top-5-accuracy: 0.9705
Epoch 31/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1785 -
accuracy: 0.7713 - top-5-accuracy: 0.9808 - val_loss: 1.0725 - val_accuracy:
0.7625 - val_top-5-accuracy: 0.9765
Epoch 32/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1736 -
accuracy: 0.7756 - top-5-accuracy: 0.9811 - val_loss: 1.0774 - val_accuracy:
0.7520 - val_top-5-accuracy: 0.9765
Epoch 33/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1743 -
accuracy: 0.7740 - top-5-accuracy: 0.9817 - val_loss: 1.0732 - val_accuracy:
0.7520 - val_top-5-accuracy: 0.9770
Epoch 34/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1758 -
accuracy: 0.7756 - top-5-accuracy: 0.9820 - val_loss: 1.0774 - val_accuracy:
0.7610 - val_top-5-accuracy: 0.9755
Epoch 35/100
336/336 [==============================] - 5s 16ms/step - loss: 1.1742 -
accuracy: 0.7749 - top-5-accuracy: 0.9820 - val_loss: 1.1192 - val_accuracy:
0.7495 - val_top-5-accuracy: 0.9640
Epoch 36/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1728 -
accuracy: 0.7760 - top-5-accuracy: 0.9811 - val_loss: 1.0592 - val_accuracy:
0.7555 - val_top-5-accuracy: 0.9765
Epoch 37/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1682 -
accuracy: 0.7779 - top-5-accuracy: 0.9820 - val_loss: 1.1028 - val_accuracy:
0.7425 - val_top-5-accuracy: 0.9710
Epoch 38/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1738 -
accuracy: 0.7756 - top-5-accuracy: 0.9821 - val_loss: 1.1035 - val_accuracy:
0.7460 - val_top-5-accuracy: 0.9710
```

```
Epoch 39/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1810 -
accuracy: 0.7730 - top-5-accuracy: 0.9817 - val_loss: 1.0542 - val_accuracy:
0.7610 - val_top-5-accuracy: 0.9780
Epoch 40/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1746 -
accuracy: 0.7773 - top-5-accuracy: 0.9806 - val_loss: 1.0781 - val_accuracy:
0.7545 - val_top-5-accuracy: 0.9730
Epoch 41/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1726 -
accuracy: 0.7766 - top-5-accuracy: 0.9821 - val_loss: 1.0752 - val_accuracy:
0.7625 - val_top-5-accuracy: 0.9680
Epoch 42/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1686 -
accuracy: 0.7792 - top-5-accuracy: 0.9816 - val_loss: 1.1061 - val_accuracy:
0.7470 - val_top-5-accuracy: 0.9690
Epoch 43/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1707 -
accuracy: 0.7763 - top-5-accuracy: 0.9824 - val_loss: 1.0594 - val_accuracy:
0.7630 - val_top-5-accuracy: 0.9760
Epoch 44/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1676 -
accuracy: 0.7805 - top-5-accuracy: 0.9823 - val_loss: 1.0902 - val_accuracy:
0.7560 - val_top-5-accuracy: 0.9790
Epoch 45/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1682 -
accuracy: 0.7802 - top-5-accuracy: 0.9818 - val_loss: 1.1229 - val_accuracy:
0.7425 - val_top-5-accuracy: 0.9765
Epoch 46/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1664 -
accuracy: 0.7797 - top-5-accuracy: 0.9819 - val_loss: 1.0773 - val_accuracy:
0.7595 - val_top-5-accuracy: 0.9740
Epoch 47/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1683 -
accuracy: 0.7777 - top-5-accuracy: 0.9809 - val_loss: 1.0712 - val_accuracy:
0.7575 - val_top-5-accuracy: 0.9730
Epoch 48/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1684 -
accuracy: 0.7775 - top-5-accuracy: 0.9817 - val_loss: 1.0917 - val_accuracy:
0.7515 - val_top-5-accuracy: 0.9710
Epoch 49/100
336/336 [==============================] - 5s 15ms/step - loss: 1.1666 -
accuracy: 0.7797 - top-5-accuracy: 0.9830 - val_loss: 1.0966 - val_accuracy:
0.7530 - val_top-5-accuracy: 0.9735
40/40 [==============================] - 0s 6ms/step - loss: 1.0897 - accuracy:
0.7524 - top-5-accuracy: 0.9754
Test accuracy: 75.24%
Test top 5 accuracy: 97.54%
```

[80]:
```
cct_classifier = create_cct_classifier()
cct_classifier.load_weights("./tmp/checkpoint_model_cct_classifier")
cct_classifier._name = 'model_cct_classifier_3'
history = run_experiment3(cct_classifier,"cct")
```

```
Epoch 1/100
336/336 [==============================] - 13s 30ms/step - loss: 1.3222 -
accuracy: 0.6973 - top-5-accuracy: 0.9716 - val_loss: 1.2028 - val_accuracy:
0.6935 - val_top-5-accuracy: 0.9680
Epoch 2/100
336/336 [==============================] - 9s 28ms/step - loss: 1.2756 -
accuracy: 0.7180 - top-5-accuracy: 0.9768 - val_loss: 1.2427 - val_accuracy:
0.6770 - val_top-5-accuracy: 0.9630
Epoch 3/100
336/336 [==============================] - 10s 29ms/step - loss: 1.2577 -
accuracy: 0.7228 - top-5-accuracy: 0.9782 - val_loss: 1.1362 - val_accuracy:
0.7185 - val_top-5-accuracy: 0.9820
Epoch 4/100
336/336 [==============================] - 9s 28ms/step - loss: 1.2403 -
accuracy: 0.7315 - top-5-accuracy: 0.9806 - val_loss: 1.1469 - val_accuracy:
0.7155 - val_top-5-accuracy: 0.9755
Epoch 5/100
336/336 [==============================] - 9s 28ms/step - loss: 1.2367 -
accuracy: 0.7343 - top-5-accuracy: 0.9794 - val_loss: 1.2146 - val_accuracy:
0.6835 - val_top-5-accuracy: 0.9735
Epoch 6/100
336/336 [==============================] - 9s 28ms/step - loss: 1.2332 -
accuracy: 0.7366 - top-5-accuracy: 0.9804 - val_loss: 1.2267 - val_accuracy:
0.6950 - val_top-5-accuracy: 0.9690
Epoch 7/100
336/336 [==============================] - 9s 27ms/step - loss: 1.2281 -
accuracy: 0.7382 - top-5-accuracy: 0.9797 - val_loss: 1.2470 - val_accuracy:
0.6815 - val_top-5-accuracy: 0.9690
Epoch 8/100
336/336 [==============================] - 9s 27ms/step - loss: 1.2292 -
accuracy: 0.7359 - top-5-accuracy: 0.9813 - val_loss: 1.2224 - val_accuracy:
0.6925 - val_top-5-accuracy: 0.9725
Epoch 9/100
336/336 [==============================] - 10s 28ms/step - loss: 1.2231 -
accuracy: 0.7409 - top-5-accuracy: 0.9804 - val_loss: 1.1608 - val_accuracy:
0.7200 - val_top-5-accuracy: 0.9725
Epoch 10/100
336/336 [==============================] - 9s 28ms/step - loss: 1.2077 -
accuracy: 0.7480 - top-5-accuracy: 0.9819 - val_loss: 1.1958 - val_accuracy:
0.6905 - val_top-5-accuracy: 0.9785
Epoch 11/100
336/336 [==============================] - 9s 27ms/step - loss: 1.2122 -
accuracy: 0.7451 - top-5-accuracy: 0.9817 - val_loss: 1.2066 - val_accuracy:
```

```
0.7000 - val_top-5-accuracy: 0.9640
Epoch 12/100
336/336 [==============================] - 9s 27ms/step - loss: 1.2148 -
accuracy: 0.7457 - top-5-accuracy: 0.9820 - val_loss: 1.2573 - val_accuracy:
0.6790 - val_top-5-accuracy: 0.9720
Epoch 13/100
336/336 [==============================] - 9s 28ms/step - loss: 1.2052 -
accuracy: 0.7497 - top-5-accuracy: 0.9810 - val_loss: 1.1566 - val_accuracy:
0.7155 - val_top-5-accuracy: 0.9720
Epoch 14/100
336/336 [==============================] - 9s 27ms/step - loss: 1.2071 -
accuracy: 0.7491 - top-5-accuracy: 0.9819 - val_loss: 1.1775 - val_accuracy:
0.7200 - val_top-5-accuracy: 0.9670
Epoch 15/100
336/336 [==============================] - 10s 28ms/step - loss: 1.2032 -
accuracy: 0.7523 - top-5-accuracy: 0.9820 - val_loss: 1.2142 - val_accuracy:
0.6990 - val_top-5-accuracy: 0.9710
Epoch 16/100
336/336 [==============================] - 9s 27ms/step - loss: 1.2020 -
accuracy: 0.7508 - top-5-accuracy: 0.9822 - val_loss: 1.2407 - val_accuracy:
0.6815 - val_top-5-accuracy: 0.9680
Epoch 17/100
336/336 [==============================] - 9s 27ms/step - loss: 1.2113 -
accuracy: 0.7465 - top-5-accuracy: 0.9815 - val_loss: 1.1886 - val_accuracy:
0.7060 - val_top-5-accuracy: 0.9725
Epoch 18/100
336/336 [==============================] - 9s 27ms/step - loss: 1.2006 -
accuracy: 0.7520 - top-5-accuracy: 0.9809 - val_loss: 1.2415 - val_accuracy:
0.6930 - val_top-5-accuracy: 0.9670
Epoch 19/100
336/336 [==============================] - 9s 27ms/step - loss: 1.2009 -
accuracy: 0.7490 - top-5-accuracy: 0.9824 - val_loss: 1.2115 - val_accuracy:
0.6905 - val_top-5-accuracy: 0.9620
Epoch 20/100
336/336 [==============================] - 9s 28ms/step - loss: 1.1943 -
accuracy: 0.7543 - top-5-accuracy: 0.9828 - val_loss: 1.1549 - val_accuracy:
0.7270 - val_top-5-accuracy: 0.9680
Epoch 21/100
336/336 [==============================] - 9s 28ms/step - loss: 1.1944 -
accuracy: 0.7561 - top-5-accuracy: 0.9828 - val_loss: 1.2176 - val_accuracy:
0.6930 - val_top-5-accuracy: 0.9695
Epoch 22/100
336/336 [==============================] - 9s 28ms/step - loss: 1.1889 -
accuracy: 0.7585 - top-5-accuracy: 0.9826 - val_loss: 1.2244 - val_accuracy:
0.6900 - val_top-5-accuracy: 0.9730
Epoch 23/100
336/336 [==============================] - 9s 27ms/step - loss: 1.1927 -
accuracy: 0.7551 - top-5-accuracy: 0.9823 - val_loss: 1.3204 - val_accuracy:
```

```
0.6705 - val_top-5-accuracy: 0.9540
Epoch 24/100
336/336 [==============================] - 9s 27ms/step - loss: 1.1897 -
accuracy: 0.7575 - top-5-accuracy: 0.9824 - val_loss: 1.2166 - val_accuracy:
0.6925 - val_top-5-accuracy: 0.9700
Epoch 25/100
336/336 [==============================] - 9s 27ms/step - loss: 1.1930 -
accuracy: 0.7560 - top-5-accuracy: 0.9824 - val_loss: 1.2145 - val_accuracy:
0.6990 - val_top-5-accuracy: 0.9725
Epoch 26/100
336/336 [==============================] - 9s 28ms/step - loss: 1.1957 -
accuracy: 0.7533 - top-5-accuracy: 0.9833 - val_loss: 1.2174 - val_accuracy:
0.7080 - val_top-5-accuracy: 0.9705
Epoch 27/100
336/336 [==============================] - 9s 28ms/step - loss: 1.1915 -
accuracy: 0.7583 - top-5-accuracy: 0.9821 - val_loss: 1.2754 - val_accuracy:
0.6820 - val_top-5-accuracy: 0.9725
40/40 [==============================] - 1s 11ms/step - loss: 1.2255 - accuracy:
0.6962 - top-5-accuracy: 0.9678
Test accuracy: 69.62%
Test top 5 accuracy: 96.78%
```

[27]:
```
resnet20_classifier = create_resnet_classifier(1)
resnet20_classifier._name = 'model_resnet20_classifier_3'
resnet20_classifier.load_weights("./tmp/checkpoint_model_resnet20_classifier")
history = run_experiment3(resnet20_classifier,"resnet20")
```

```
2022-03-23 20:27:49.994652: I tensorflow/stream_executor/cuda/cuda_dnn.cc:368]
Loaded cuDNN version 8200
```

```
Epoch 1/100
336/336 [==============================] - 9s 19ms/step - loss: 1.3970 -
accuracy: 0.6735 - top-5-accuracy: 0.9672 - val_loss: 1.9455 - val_accuracy:
0.4945 - val_top-5-accuracy: 0.9295
Epoch 2/100
336/336 [==============================] - 6s 17ms/step - loss: 1.3415 -
accuracy: 0.6902 - top-5-accuracy: 0.9715 - val_loss: 1.6301 - val_accuracy:
0.5290 - val_top-5-accuracy: 0.9120
Epoch 3/100
336/336 [==============================] - 6s 17ms/step - loss: 1.3213 -
accuracy: 0.7005 - top-5-accuracy: 0.9738 - val_loss: 1.5574 - val_accuracy:
0.5465 - val_top-5-accuracy: 0.9035
Epoch 4/100
336/336 [==============================] - 6s 17ms/step - loss: 1.3178 -
accuracy: 0.7030 - top-5-accuracy: 0.9734 - val_loss: 1.5282 - val_accuracy:
0.5950 - val_top-5-accuracy: 0.9160
Epoch 5/100
336/336 [==============================] - 6s 17ms/step - loss: 1.3134 -
accuracy: 0.7042 - top-5-accuracy: 0.9737 - val_loss: 1.5379 - val_accuracy:
```

```
0.5715 - val_top-5-accuracy: 0.9185
Epoch 6/100
336/336 [==============================] - 6s 17ms/step - loss: 1.3011 -
accuracy: 0.7087 - top-5-accuracy: 0.9751 - val_loss: 1.4872 - val_accuracy:
0.5890 - val_top-5-accuracy: 0.9450
Epoch 7/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2984 -
accuracy: 0.7141 - top-5-accuracy: 0.9751 - val_loss: 1.5668 - val_accuracy:
0.5840 - val_top-5-accuracy: 0.9270
Epoch 8/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2933 -
accuracy: 0.7134 - top-5-accuracy: 0.9749 - val_loss: 1.5004 - val_accuracy:
0.5855 - val_top-5-accuracy: 0.9465
Epoch 9/100
336/336 [==============================] - 6s 18ms/step - loss: 1.2936 -
accuracy: 0.7126 - top-5-accuracy: 0.9752 - val_loss: 1.3911 - val_accuracy:
0.6335 - val_top-5-accuracy: 0.9575
Epoch 10/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2902 -
accuracy: 0.7132 - top-5-accuracy: 0.9750 - val_loss: 1.6177 - val_accuracy:
0.5550 - val_top-5-accuracy: 0.9190
Epoch 11/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2861 -
accuracy: 0.7167 - top-5-accuracy: 0.9757 - val_loss: 1.3748 - val_accuracy:
0.6255 - val_top-5-accuracy: 0.9465
Epoch 12/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2801 -
accuracy: 0.7175 - top-5-accuracy: 0.9768 - val_loss: 1.8656 - val_accuracy:
0.5085 - val_top-5-accuracy: 0.9045
Epoch 13/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2869 -
accuracy: 0.7144 - top-5-accuracy: 0.9753 - val_loss: 1.6347 - val_accuracy:
0.5375 - val_top-5-accuracy: 0.9165
Epoch 14/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2763 -
accuracy: 0.7207 - top-5-accuracy: 0.9755 - val_loss: 1.4502 - val_accuracy:
0.6025 - val_top-5-accuracy: 0.9365
Epoch 15/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2807 -
accuracy: 0.7177 - top-5-accuracy: 0.9756 - val_loss: 1.5019 - val_accuracy:
0.6020 - val_top-5-accuracy: 0.9200
Epoch 16/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2809 -
accuracy: 0.7192 - top-5-accuracy: 0.9755 - val_loss: 1.6117 - val_accuracy:
0.5525 - val_top-5-accuracy: 0.9245
Epoch 17/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2791 -
accuracy: 0.7168 - top-5-accuracy: 0.9756 - val_loss: 1.5857 - val_accuracy:
```

```
0.5655 - val_top-5-accuracy: 0.9260
Epoch 18/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2794 -
accuracy: 0.7182 - top-5-accuracy: 0.9748 - val_loss: 1.6208 - val_accuracy:
0.5485 - val_top-5-accuracy: 0.9605
Epoch 19/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2745 -
accuracy: 0.7210 - top-5-accuracy: 0.9762 - val_loss: 1.4754 - val_accuracy:
0.5890 - val_top-5-accuracy: 0.9375
Epoch 20/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2738 -
accuracy: 0.7180 - top-5-accuracy: 0.9762 - val_loss: 1.6899 - val_accuracy:
0.5455 - val_top-5-accuracy: 0.8860
Epoch 21/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2688 -
accuracy: 0.7223 - top-5-accuracy: 0.9759 - val_loss: 1.4081 - val_accuracy:
0.6100 - val_top-5-accuracy: 0.9470
Epoch 22/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2816 -
accuracy: 0.7176 - top-5-accuracy: 0.9756 - val_loss: 1.4079 - val_accuracy:
0.6010 - val_top-5-accuracy: 0.9250
Epoch 23/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2741 -
accuracy: 0.7212 - top-5-accuracy: 0.9754 - val_loss: 1.5706 - val_accuracy:
0.5720 - val_top-5-accuracy: 0.9125
Epoch 24/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2746 -
accuracy: 0.7199 - top-5-accuracy: 0.9757 - val_loss: 1.5677 - val_accuracy:
0.5555 - val_top-5-accuracy: 0.9295
Epoch 25/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2720 -
accuracy: 0.7217 - top-5-accuracy: 0.9763 - val_loss: 1.3978 - val_accuracy:
0.6120 - val_top-5-accuracy: 0.9365
Epoch 26/100
336/336 [==============================] - 6s 17ms/step - loss: 1.2699 -
accuracy: 0.7221 - top-5-accuracy: 0.9761 - val_loss: 1.5659 - val_accuracy:
0.5535 - val_top-5-accuracy: 0.9145
40/40 [==============================] - 0s 8ms/step - loss: 1.4115 - accuracy:
0.6130 - top-5-accuracy: 0.9426
Test accuracy: 61.3%
Test top 5 accuracy: 94.26%
```

[82]:
```
resnet44_classifier = create_resnet_classifier(7)
resnet44_classifier._name = 'model_resnet44_classifier_3'
resnet44_classifier.load_weights("./tmp/checkpoint_model_resnet44_classifier")
history = run_experiment3(resnet44_classifier,"resnet44")
```

```
Epoch 1/100
```

```
336/336 [==============================] - 29s 70ms/step - loss: 1.3184 -
accuracy: 0.7261 - top-5-accuracy: 0.9641 - val_loss: 1.0387 - val_accuracy:
0.7840 - val_top-5-accuracy: 0.9810
Epoch 2/100
336/336 [==============================] - 23s 68ms/step - loss: 1.2688 -
accuracy: 0.7439 - top-5-accuracy: 0.9722 - val_loss: 1.0833 - val_accuracy:
0.7470 - val_top-5-accuracy: 0.9710
Epoch 3/100
336/336 [==============================] - 23s 67ms/step - loss: 1.2502 -
accuracy: 0.7510 - top-5-accuracy: 0.9726 - val_loss: 1.0187 - val_accuracy:
0.7825 - val_top-5-accuracy: 0.9705
Epoch 4/100
336/336 [==============================] - 23s 67ms/step - loss: 1.2320 -
accuracy: 0.7606 - top-5-accuracy: 0.9732 - val_loss: 1.2045 - val_accuracy:
0.7070 - val_top-5-accuracy: 0.9720
Epoch 5/100
336/336 [==============================] - 22s 67ms/step - loss: 1.2231 -
accuracy: 0.7591 - top-5-accuracy: 0.9737 - val_loss: 1.0986 - val_accuracy:
0.7350 - val_top-5-accuracy: 0.9715
Epoch 6/100
336/336 [==============================] - 23s 67ms/step - loss: 1.2187 -
accuracy: 0.7649 - top-5-accuracy: 0.9749 - val_loss: 1.0620 - val_accuracy:
0.7620 - val_top-5-accuracy: 0.9750
Epoch 7/100
336/336 [==============================] - 23s 68ms/step - loss: 1.2144 -
accuracy: 0.7688 - top-5-accuracy: 0.9753 - val_loss: 1.0742 - val_accuracy:
0.7505 - val_top-5-accuracy: 0.9700
Epoch 8/100
336/336 [==============================] - 23s 67ms/step - loss: 1.2002 -
accuracy: 0.7742 - top-5-accuracy: 0.9765 - val_loss: 1.0460 - val_accuracy:
0.7565 - val_top-5-accuracy: 0.9815
Epoch 9/100
336/336 [==============================] - 23s 67ms/step - loss: 1.2002 -
accuracy: 0.7745 - top-5-accuracy: 0.9759 - val_loss: 1.0995 - val_accuracy:
0.7400 - val_top-5-accuracy: 0.9735
Epoch 10/100
336/336 [==============================] - 22s 66ms/step - loss: 1.1984 -
accuracy: 0.7737 - top-5-accuracy: 0.9768 - val_loss: 1.0719 - val_accuracy:
0.7520 - val_top-5-accuracy: 0.9725
Epoch 11/100
336/336 [==============================] - 23s 67ms/step - loss: 1.1954 -
accuracy: 0.7753 - top-5-accuracy: 0.9761 - val_loss: 1.1420 - val_accuracy:
0.7300 - val_top-5-accuracy: 0.9770
Epoch 12/100
336/336 [==============================] - 23s 67ms/step - loss: 1.1902 -
accuracy: 0.7772 - top-5-accuracy: 0.9775 - val_loss: 1.2003 - val_accuracy:
0.7080 - val_top-5-accuracy: 0.9650
Epoch 13/100
```

```
336/336 [==============================] - 23s 68ms/step - loss: 1.1868 -
accuracy: 0.7787 - top-5-accuracy: 0.9773 - val_loss: 1.1550 - val_accuracy:
0.7270 - val_top-5-accuracy: 0.9670
Epoch 14/100
336/336 [==============================] - 22s 66ms/step - loss: 1.1836 -
accuracy: 0.7786 - top-5-accuracy: 0.9776 - val_loss: 1.1601 - val_accuracy:
0.7270 - val_top-5-accuracy: 0.9530
Epoch 15/100
336/336 [==============================] - 23s 67ms/step - loss: 1.1811 -
accuracy: 0.7831 - top-5-accuracy: 0.9772 - val_loss: 1.0726 - val_accuracy:
0.7465 - val_top-5-accuracy: 0.9790
Epoch 16/100
336/336 [==============================] - 23s 67ms/step - loss: 1.1800 -
accuracy: 0.7850 - top-5-accuracy: 0.9775 - val_loss: 1.0773 - val_accuracy:
0.7625 - val_top-5-accuracy: 0.9755
Epoch 17/100
336/336 [==============================] - 22s 66ms/step - loss: 1.1746 -
accuracy: 0.7846 - top-5-accuracy: 0.9785 - val_loss: 0.9828 - val_accuracy:
0.7960 - val_top-5-accuracy: 0.9845
Epoch 18/100
336/336 [==============================] - 22s 67ms/step - loss: 1.1803 -
accuracy: 0.7812 - top-5-accuracy: 0.9769 - val_loss: 1.1136 - val_accuracy:
0.7345 - val_top-5-accuracy: 0.9705
Epoch 19/100
336/336 [==============================] - 22s 66ms/step - loss: 1.1787 -
accuracy: 0.7848 - top-5-accuracy: 0.9769 - val_loss: 1.0786 - val_accuracy:
0.7580 - val_top-5-accuracy: 0.9710
Epoch 20/100
336/336 [==============================] - 23s 68ms/step - loss: 1.1706 -
accuracy: 0.7865 - top-5-accuracy: 0.9779 - val_loss: 1.0587 - val_accuracy:
0.7790 - val_top-5-accuracy: 0.9690
Epoch 21/100
336/336 [==============================] - 22s 66ms/step - loss: 1.1713 -
accuracy: 0.7851 - top-5-accuracy: 0.9775 - val_loss: 1.1209 - val_accuracy:
0.7395 - val_top-5-accuracy: 0.9745
Epoch 22/100
336/336 [==============================] - 23s 67ms/step - loss: 1.1650 -
accuracy: 0.7888 - top-5-accuracy: 0.9775 - val_loss: 1.0538 - val_accuracy:
0.7785 - val_top-5-accuracy: 0.9720
Epoch 23/100
336/336 [==============================] - 23s 68ms/step - loss: 1.1595 -
accuracy: 0.7907 - top-5-accuracy: 0.9773 - val_loss: 1.0356 - val_accuracy:
0.7760 - val_top-5-accuracy: 0.9795
Epoch 24/100
336/336 [==============================] - 22s 66ms/step - loss: 1.1674 -
accuracy: 0.7884 - top-5-accuracy: 0.9791 - val_loss: 1.0394 - val_accuracy:
0.7750 - val_top-5-accuracy: 0.9775
Epoch 25/100
```

```
336/336 [==============================] - 23s 68ms/step - loss: 1.1576 -
accuracy: 0.7928 - top-5-accuracy: 0.9781 - val_loss: 1.1575 - val_accuracy:
0.7270 - val_top-5-accuracy: 0.9725
Epoch 26/100
336/336 [==============================] - 23s 67ms/step - loss: 1.1601 -
accuracy: 0.7916 - top-5-accuracy: 0.9777 - val_loss: 1.0576 - val_accuracy:
0.7685 - val_top-5-accuracy: 0.9645
Epoch 27/100
336/336 [==============================] - 23s 68ms/step - loss: 1.1563 -
accuracy: 0.7942 - top-5-accuracy: 0.9777 - val_loss: 1.1093 - val_accuracy:
0.7500 - val_top-5-accuracy: 0.9625
Epoch 28/100
336/336 [==============================] - 22s 66ms/step - loss: 1.1611 -
accuracy: 0.7886 - top-5-accuracy: 0.9779 - val_loss: 1.1250 - val_accuracy:
0.7285 - val_top-5-accuracy: 0.9700
Epoch 29/100
336/336 [==============================] - 22s 66ms/step - loss: 1.1614 -
accuracy: 0.7907 - top-5-accuracy: 0.9788 - val_loss: 1.1426 - val_accuracy:
0.7365 - val_top-5-accuracy: 0.9670
Epoch 30/100
336/336 [==============================] - 22s 67ms/step - loss: 1.1606 -
accuracy: 0.7916 - top-5-accuracy: 0.9785 - val_loss: 0.9757 - val_accuracy:
0.7975 - val_top-5-accuracy: 0.9840
Epoch 31/100
336/336 [==============================] - 22s 66ms/step - loss: 1.1526 -
accuracy: 0.7939 - top-5-accuracy: 0.9780 - val_loss: 1.1393 - val_accuracy:
0.7365 - val_top-5-accuracy: 0.9635
Epoch 32/100
336/336 [==============================] - 22s 67ms/step - loss: 1.1530 -
accuracy: 0.7927 - top-5-accuracy: 0.9794 - val_loss: 1.1019 - val_accuracy:
0.7555 - val_top-5-accuracy: 0.9710
40/40 [==============================] - 1s 17ms/step - loss: 1.0990 - accuracy:
0.7552 - top-5-accuracy: 0.9604
Test accuracy: 75.52%
Test top 5 accuracy: 96.04%
```

[274]:
```
# [BUILD A MORE SOPHISTICATED PREDICTIVE MODEL]

# write your code here...
vgg_model = create_vgg_classifier(3)
vgg_model.load_weights("./tmp/checkpoint_model_vgg_classifier")

def model_I(image):
    '''
    This function should takes in the image of dimension 32*32*3 as input and␣
 ↪returns a label prediction
    '''
    img_prep = prep_pixel(image)
    return vgg_model.predict(img_prep)

    # write your code here...
```

[275]:
```
# [ADD WEAKLY SUPERVISED LEARNING FEATURE TO MODEL I]

# write your code here...
vgg_model = create_vgg_classifier(3)
vgg_model.load_weights("./tmp/checkpoint_model_vgg_classifier_2")

def model_II(image):
    '''
    This function should takes in the image of dimension 32*32*3 as input and␣
 ↪returns a label prediction
    '''
    img_prep = prep_pixel(image)
    return vgg_model.predict(img_prep)
    # write your code here...
```

1.8 3. Evaluation

For assessment, we will evaluate your final model on a hidden test dataset with clean labels by the `evaluation` function defined as follows. Although you will not have the access to the test set, the function would be useful for the model developments. For example, you can split the small training set, using one portion for weakly supervised learning and the other for validation purpose.

[29]:
```
# [DO NOT MODIFY THIS CELL]
def evaluation(model, test_labels, test_imgs):
    y_true = test_labels
    y_pred = []
    for image in test_imgs:
        y_pred.append(model(image))
    print(classification_report(y_true, y_pred))
```

[30]:
```
# [DO NOT MODIFY THIS CELL]
# This is the code for evaluating the prediction performance on a testset
# You will get an error if running this cell, as you do not have the testset
# Nonetheless, you can create your own validation set to run the evlauation
n_test = 10000
test_labels = np.genfromtxt('../data/test_labels.csv', delimiter=',',␣
 ↪dtype="int8")
test_imgs = np.empty((n_test,32,32,3))
for i in range(n_test):
    img_fn = f'../data/test_images/test{i+1:05d}.png'
    test_imgs[i,:,:,:]=cv2.cvtColor(cv2.imread(img_fn),cv2.COLOR_BGR2RGB)
evaluation(baseline_model, test_labels, test_imgs)
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
/tmp/ipykernel_7700/1258430874.py in <module>
      4 # Nonetheless, you can create your own validation set to run the␣
 ↪evlauation
      5 n_test = 10000
----> 6 test_labels = np.genfromtxt('../data/test_labels.csv', delimiter=',',␣
 ↪dtype="int8")
      7 test_imgs = np.empty((n_test,32,32,3))
      8 for i in range(n_test):

/opt/conda/lib/python3.7/site-packages/numpy/lib/npyio.py in genfromtxt(fname,␣
 ↪dtype, comments, delimiter, skip_header, skip_footer, converters,␣
 ↪missing_values, filling_values, usecols, names, excludelist, deletechars,␣
 ↪replace_space, autostrip, case_sensitive, defaultfmt, unpack, usemask, loose,␣
 ↪invalid_raise, max_rows, encoding)
   1747             fname = os_fspath(fname)
   1748         if isinstance(fname, str):
-> 1749             fid = np.lib._datasource.open(fname, 'rt', encoding=encodin
   1750             fid_ctx = contextlib.closing(fid)
   1751         else:

/opt/conda/lib/python3.7/site-packages/numpy/lib/_datasource.py in open(path,␣
 ↪mode, destpath, encoding, newline)
    193
    194     ds = DataSource(destpath)
--> 195     return ds.open(path, mode, encoding=encoding, newline=newline)
    196
    197

/opt/conda/lib/python3.7/site-packages/numpy/lib/_datasource.py in open(self,␣
 ↪path, mode, encoding, newline)
    533                                       encoding=encoding, newline=newlin
    534         else:
--> 535             raise IOError("%s not found." % path)
    536
    537

OSError: ../data/test_labels.csv not found.
```

The overall accuracy is 0.24, which is better than random guess (which should have a accuracy around 0.10). For the project, you should try to improve the performance by the following strategies:

- Consider a better choice of model architectures, hyperparameters, or training scheme for the predictive model;
- Use both `clean_noisy_trainset` and `noisy_trainset` for model training via **weakly supervised learning** methods. One possible solution is to train a "label-correction" model using the former, correct the labels in the latter, and train the final predictive model using

the corrected dataset.

- Apply techniques such as k-fold cross validation to avoid overfitting;
- Any other reasonable strategies.