Loading the Dataset and Processing it.

The Mnist Fashion Dataset has been loaded from the keras datasets and have been normalized by dividing with 255.

```
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from keras.utils import to_categorical
fashion_mnist = keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

x_train = x_train.astype("float32") / 255
x_test = x_test.astype("float32") / 255

x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
test_size=0.2, random_state=42)

print("x_train shape:", x_train.shape)
print("x_val shape:", x_val.shape)
print("x_test shape:", x_test.shape)
print("y_train shape:", y_train.shape)
print("y_val shape:", y_val.shape)
print("y_test shape:", y_test.shape)
```

```
2025-04-27 13:49:15.185344: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to
register cuFFT factory: Attempting to register factory for plugin
cuFFT when one has already been registered
WARNING: All log messages before absl::InitializeLog() is called are
written to STDERR
E0000 00:00:1745761755.418464      31 cuda_dnn.cc:8310] Unable to
register cuDNN factory: Attempting to register factory for plugin
cuDNN when one has already been registered
E0000 00:00:1745761755.490230      31 cuda_blas.cc:1418] Unable to
register cuBLAS factory: Attempting to register factory for plugin
cuBLAS when one has already been registered
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
x_train shape: (48000, 28, 28)
x_val shape: (12000, 28, 28)
x_test shape: (10000, 28, 28)
y_train shape: (48000,)
y_val shape: (12000,)
y_test shape: (10000,)
```

Creating the Residual Layer for ResNet 18 and ResNet 34

```
import tensorflow
import tensorflow.keras as keras

class ResidualUnit(keras.layers.Layer):
    def __init__(self, filters, strides=1, activation="relu",
**kwargs):
        super().__init__(**kwargs)
        self.activation = keras.activations.get(activation)
        self.main_layers = [
        keras.layers.Conv2D(filters, 3, strides=strides,
padding="same", use_bias=False),
        keras.layers.BatchNormalization(),
        self.activation,
        keras.layers.Conv2D(filters, 3, strides=1, padding="same",
use_bias=False),
        keras.layers.BatchNormalization()]
        self.skip_layers = []
        if strides > 1:
            # If the input and output shapes are different, we need to
adjust the skip connection
            # to match the output shape of the main path.
            # This is done by applying a 1x1 convolution to the input.
            # The 1x1 convolution will change the number of channels
to match the output of the main path.
            # The strides argument is used to downsample the input.
            # The padding argument is set to "same" to ensure that the
output shape matches the main path.
            # The use_bias argument is set to False because we already
have a batch normalization layer after the convolution.
            # The batch normalization layer is used to normalize the
output of the convolution.
            self.skip_layers = [
                keras.layers.Conv2D(filters,1,
strides=strides,padding="same", use_bias=False),
                keras.layers.BatchNormalization()]
    def call(self, inputs):
        Z = inputs
        for layer in self.main_layers:
            Z = layer(Z)
    # Skip connection
        skip_Z = inputs
        for layer in self.skip_layers:
            skip_Z = layer(skip_Z)
        return self.activation(Z + skip_Z)
```

ResNet - 18

```
from keras import models
from keras import layers
model = keras.models.Sequential()
model.add(keras.layers.Resizing(224, 224,
interpolation="bilinear",input_shape=[28,28,1]))
model.add(layers.Conv2D(64, (7,7), strides=2, padding='same',
use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.ReLU())
model.add(layers.MaxPooling2D((3, 3), strides=2, padding='same'))

Prv = 64

for filters in [64] * 2 + [128] * 2 + [256] * 2 + [512] * 2:
    strides = 1 if filters == Prv else 2
    model.add(ResidualUnit(filters, strides=strides))
    Prv = filters

model.add(layers.GlobalAveragePooling2D())
model.add(layers.Flatten())
model.add(layers.Dense(10, activation='softmax'))

model.summary()


model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
history=model.fit(x_train,y_train, epochs=15, batch_size=32,
validation_data=(x_val,y_val))


#Print the training, validation and test accuracy

train_loss, train_accuracy = model.evaluate(x_train, y_train)
```

```
val_loss, val_accuracy = model.evaluate(x_val, y_val)
test_loss, test_accuracy = model.evaluate(x_test, y_test)

print("\nEvaluation Results:")
print("Training Loss: {:.4f}  Training Accuracy:
{:.4f}".format(train_loss, train_accuracy))
print("Validation Loss: {:.4f}  Validation Accuracy:
{:.4f}".format(val_loss, val_accuracy))
print("Test Loss: {:.4f}  Test Accuracy: {:.4f}".format(test_loss,
test_accuracy))

#Plot the training and validation accuracy and loss
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
import matplotlib.pyplot as plt
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

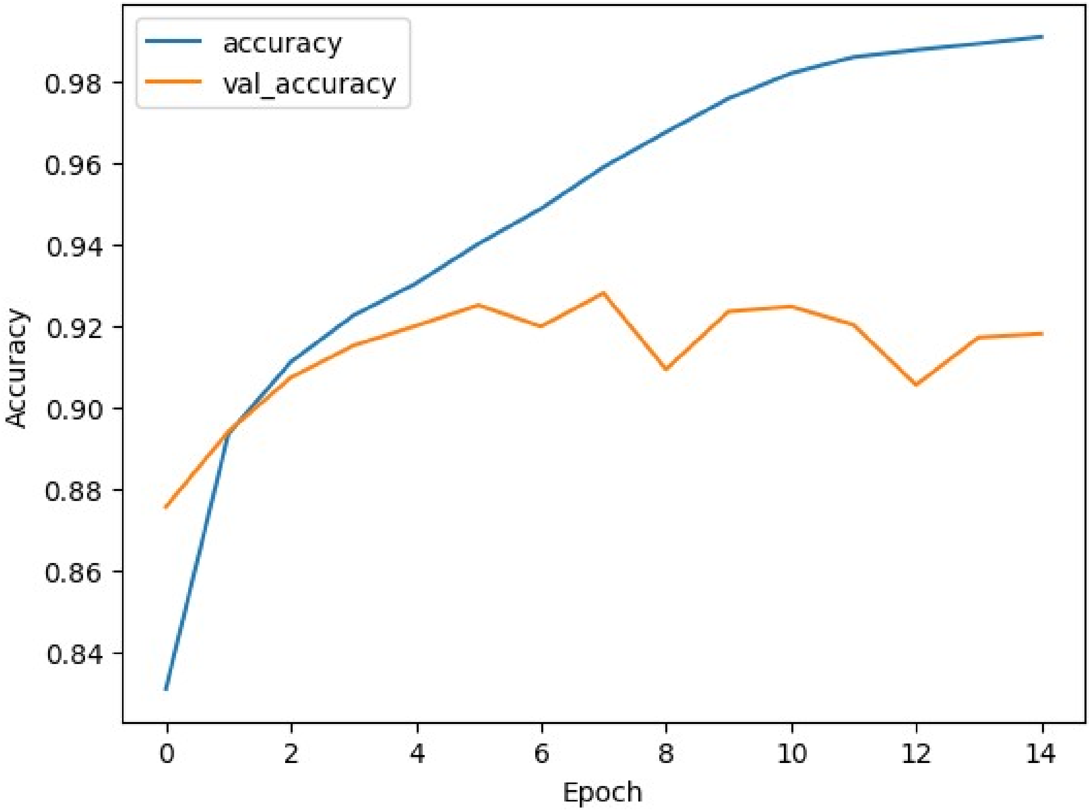
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
preprocessing/tf_data_layer.py:19: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
I0000 00:00:1745762202.548321      31 gpu_device.cc:2022] Created
device /job:localhost/replica:0/task:0/device:GPU:0 with 13942 MB
memory:  -> device: 0, name: Tesla T4, pci bus id: 0000:00:04.0,
compute capability: 7.5
I0000 00:00:1745762202.549059      31 gpu_device.cc:2022] Created
device /job:localhost/replica:0/task:0/device:GPU:1 with 13942 MB
memory:  -> device: 1, name: Tesla T4, pci bus id: 0000:00:05.0,
compute capability: 7.5

Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
```

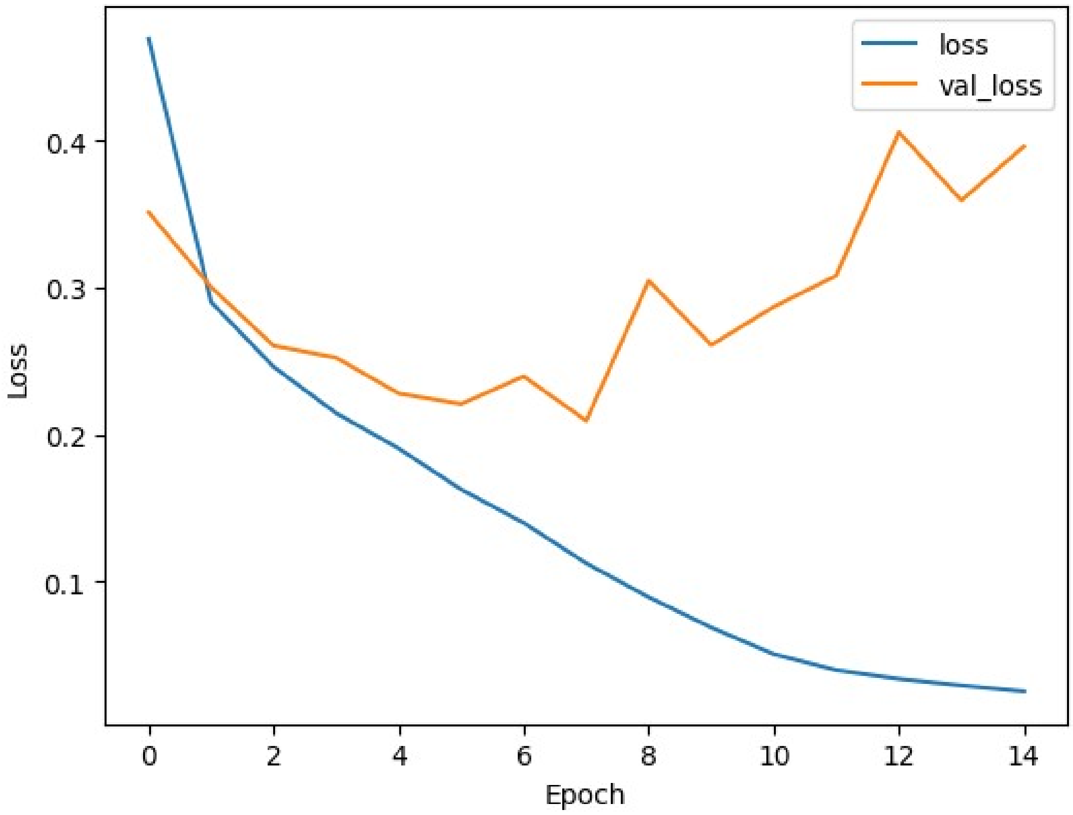
```
━━━━━━━━━━━━━━━━━┩
│ resizing (Resizing)                  │ (None, 224, 224, 1)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d (Conv2D)                      │ (None, 112, 112, 64)        │
3,136 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization                  │ (None, 112, 112, 64)        │
256 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ re_lu (ReLU)                         │ (None, 112, 112, 64)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 56, 56, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit (ResidualUnit)         │ (None, 56, 56, 64)          │
74,240 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_1 (ResidualUnit)       │ (None, 56, 56, 64)          │
74,240 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_2 (ResidualUnit)       │ (None, 28, 28, 128)         │
230,912 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_3 (ResidualUnit)       │ (None, 28, 28, 128)         │
295,936 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_4 (ResidualUnit)       │ (None, 14, 14, 256)         │
920,576 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_5 (ResidualUnit)       │ (None, 14, 14, 256)         │
1,181,696 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_6 (ResidualUnit)       │ (None, 7, 7, 512)           │
3,676,160 │
```

```
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_7 (ResidualUnit)       │ (None, 7, 7, 512)           │
4,722,688 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ global_average_pooling2d             │ (None, 512)                 │
0 │
│ (GlobalAveragePooling2D)             │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ flatten (Flatten)                    │ (None, 512)                 │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense (Dense)                        │ (None, 10)                  │
5,130 │
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 11,184,970 (42.67 MB)

 Trainable params: 11,175,370 (42.63 MB)

 Non-trainable params: 9,600 (37.50 KB)

Epoch 1/15

WARNING: All log messages before absl::InitializeLog() is called are
written to STDERR
I0000 00:00:1745762220.200597      95 service.cc:148] XLA service
0x78a26400e400 initialized for platform CUDA (this does not guarantee
that XLA will be used). Devices:
I0000 00:00:1745762220.202452      95 service.cc:156]   StreamExecutor
device (0): Tesla T4, Compute Capability 7.5
I0000 00:00:1745762220.202472      95 service.cc:156]   StreamExecutor
device (1): Tesla T4, Compute Capability 7.5
I0000 00:00:1745762221.512160      95 cuda_dnn.cc:529] Loaded cuDNN
version 90300

   1/1500 ━━━━━━━━━━━━━━━━━━━━ 10:06:01 24s/step - accuracy: 0.1250 -
loss: 3.2215

I0000 00:00:1745762230.933416      95 device_compiler.h:188] Compiled
cluster using XLA!  This line is logged at most once for the lifetime
of the process.

1500/1500 ━━━━━━━━━━━━━━━━━━━━ 190s 111ms/step - accuracy: 0.7675 -
loss: 0.6707 - val_accuracy: 0.8757 - val_loss: 0.3511
Epoch 2/15
```

```
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 114ms/step - accuracy: 0.8897 -
loss: 0.2988 - val_accuracy: 0.8942 - val_loss: 0.3002
Epoch 3/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 114ms/step - accuracy: 0.9116 -
loss: 0.2459 - val_accuracy: 0.9074 - val_loss: 0.2603
Epoch 4/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 114ms/step - accuracy: 0.9212 -
loss: 0.2170 - val_accuracy: 0.9153 - val_loss: 0.2522
Epoch 5/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 113ms/step - accuracy: 0.9317 -
loss: 0.1846 - val_accuracy: 0.9201 - val_loss: 0.2278
Epoch 6/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 114ms/step - accuracy: 0.9419 -
loss: 0.1577 - val_accuracy: 0.9252 - val_loss: 0.2207
Epoch 7/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 113ms/step - accuracy: 0.9489 -
loss: 0.1420 - val_accuracy: 0.9199 - val_loss: 0.2395
Epoch 8/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 169s 113ms/step - accuracy: 0.9610 -
loss: 0.1076 - val_accuracy: 0.9281 - val_loss: 0.2092
Epoch 9/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 114ms/step - accuracy: 0.9705 -
loss: 0.0820 - val_accuracy: 0.9094 - val_loss: 0.3045
Epoch 10/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 113ms/step - accuracy: 0.9784 -
loss: 0.0616 - val_accuracy: 0.9236 - val_loss: 0.2608
Epoch 11/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 113ms/step - accuracy: 0.9842 -
loss: 0.0434 - val_accuracy: 0.9248 - val_loss: 0.2869
Epoch 12/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 114ms/step - accuracy: 0.9886 -
loss: 0.0319 - val_accuracy: 0.9203 - val_loss: 0.3081
Epoch 13/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 113ms/step - accuracy: 0.9902 -
loss: 0.0268 - val_accuracy: 0.9056 - val_loss: 0.4058
Epoch 14/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 113ms/step - accuracy: 0.9913 -
loss: 0.0238 - val_accuracy: 0.9172 - val_loss: 0.3593
Epoch 15/15
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 170s 113ms/step - accuracy: 0.9922 -
loss: 0.0215 - val_accuracy: 0.9182 - val_loss: 0.3960
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 37s 25ms/step - accuracy: 0.9856 -
loss: 0.0410
375/375 ━━━━━━━━━━━━━━━━━━━━ 9s 25ms/step - accuracy: 0.9201 - loss:
0.3760
313/313 ━━━━━━━━━━━━━━━━━━━━ 12s 31ms/step - accuracy: 0.9164 - loss:
0.4202

Evaluation Results:
```

```
Training Loss: 0.0427  Training Accuracy: 0.9849
Validation Loss: 0.3960  Validation Accuracy: 0.9182
Test Loss: 0.4169  Test Accuracy: 0.9188
```





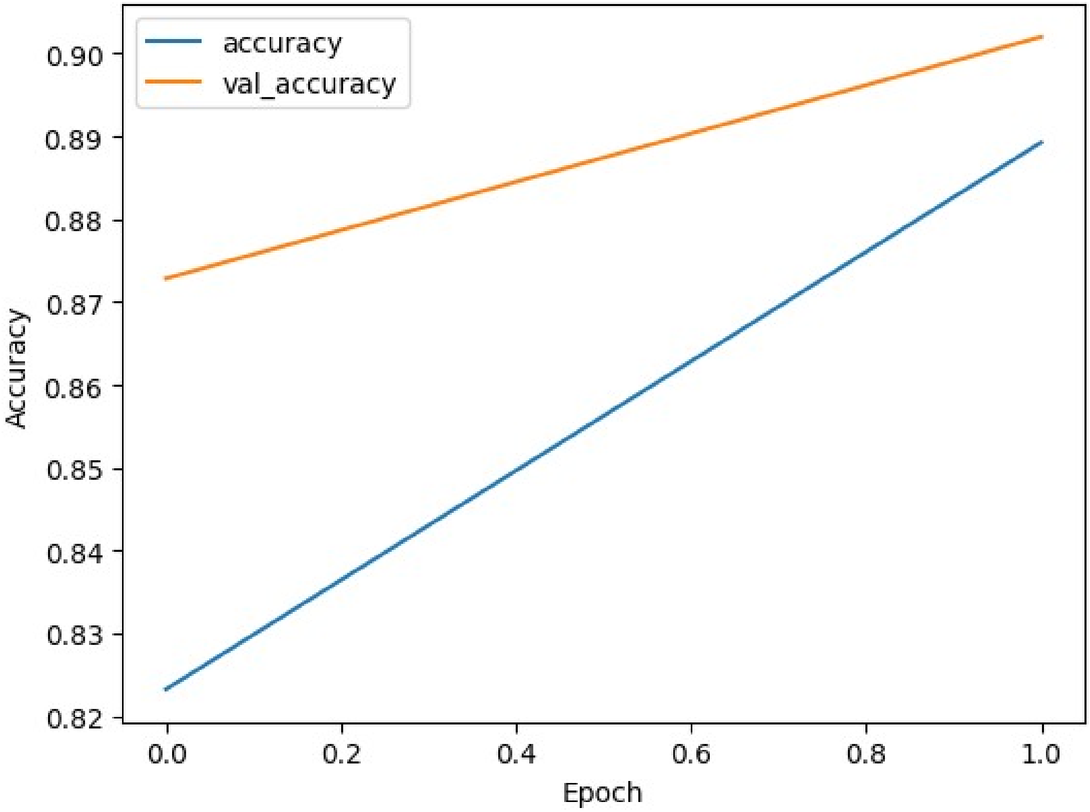
ResNet34

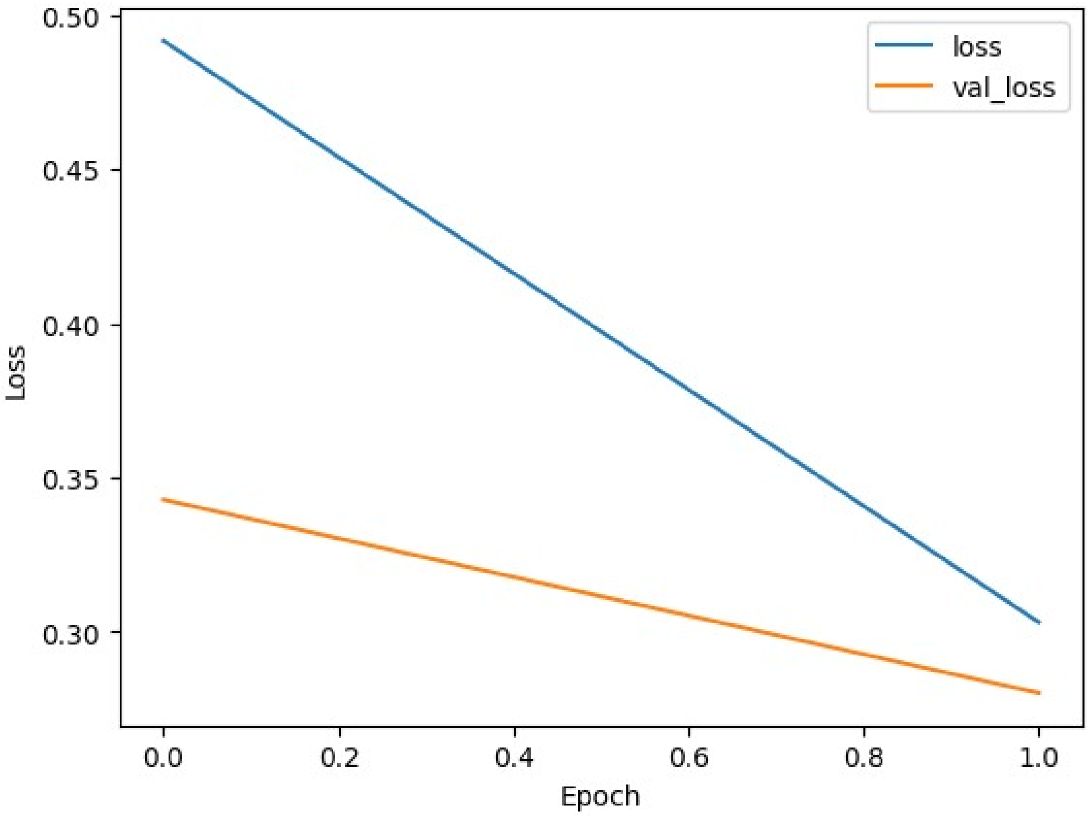
```
from keras import models
from keras import layers
model = keras.models.Sequential()
model.add(keras.layers.Resizing(224, 224,
interpolation="bilinear",input_shape=[28,28,1]))
model.add(layers.Conv2D(64, (7,7), strides=2, padding='same',
use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.ReLU())
model.add(layers.MaxPooling2D((3, 3), strides=2, padding='same'))

Prv = 64

for filters in [64] * 3 + [128] * 4 + [256] * 6 + [512] * 3:
    strides = 1 if filters == Prv else 2
    model.add(ResidualUnit(filters, strides=strides))
    Prv = filters

model.add(layers.GlobalAveragePooling2D())
model.add(layers.Flatten())
model.add(layers.Dense(10, activation='softmax'))
model.summary()


model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
history=model.fit(x_train,y_train, epochs=15, batch_size=32,
validation_data=(x_val,y_val))

#Print the training, validation and test accuracy
#Print the training, validation and test accuracy

train_loss, train_accuracy = model.evaluate(x_train, y_train)
val_loss, val_accuracy = model.evaluate(x_val, y_val)
test_loss, test_accuracy = model.evaluate(x_test, y_test)

print("\nEvaluation Results:")
print("Training Loss: {:.4f}  Training Accuracy:
{:.4f}".format(train_loss, train_accuracy))
print("Validation Loss: {:.4f}  Validation Accuracy:
{:.4f}".format(val_loss, val_accuracy))
print("Test Loss: {:.4f}  Test Accuracy: {:.4f}".format(test_loss,
test_accuracy))

import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
import matplotlib.pyplot as plt
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Model: "sequential_1"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                          ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
━━━━━━━━━━━━━━━━━┩
│ resizing_1 (Resizing)                 │ (None, 224, 224, 1)         │
0 │
├───────────────────────────────────────┼─────────────────────────────┼
```

```
─────────────────┤
│ conv2d_20 (Conv2D)                   │ (None, 112, 112, 64)        │
3,136 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_20               │ (None, 112, 112, 64)        │
256 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ re_lu_1 (ReLU)                       │ (None, 112, 112, 64)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 56, 56, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_8 (ResidualUnit)       │ (None, 56, 56, 64)          │
74,240 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_9 (ResidualUnit)       │ (None, 56, 56, 64)          │
74,240 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_10 (ResidualUnit)      │ (None, 56, 56, 64)          │
74,240 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_11 (ResidualUnit)      │ (None, 28, 28, 128)         │
230,912 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_12 (ResidualUnit)      │ (None, 28, 28, 128)         │
295,936 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_13 (ResidualUnit)      │ (None, 28, 28, 128)         │
295,936 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_14 (ResidualUnit)      │ (None, 28, 28, 128)         │
295,936 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_15 (ResidualUnit)      │ (None, 14, 14, 256)         │
920,576 │
```

```
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ residual_unit_16 (ResidualUnit)      │ (None, 14, 14, 256)         │
1,181,696 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ residual_unit_17 (ResidualUnit)      │ (None, 14, 14, 256)         │
1,181,696 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ residual_unit_18 (ResidualUnit)      │ (None, 14, 14, 256)         │
1,181,696 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ residual_unit_19 (ResidualUnit)      │ (None, 14, 14, 256)         │
1,181,696 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ residual_unit_20 (ResidualUnit)      │ (None, 14, 14, 256)         │
1,181,696 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ residual_unit_21 (ResidualUnit)      │ (None, 7, 7, 512)           │
3,676,160 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ residual_unit_22 (ResidualUnit)      │ (None, 7, 7, 512)           │
4,722,688 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ residual_unit_23 (ResidualUnit)      │ (None, 7, 7, 512)           │
4,722,688 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ global_average_pooling2d_1           │ (None, 512)                 │
0 │
│ (GlobalAveragePooling2D)             │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ flatten_1 (Flatten)                  │ (None, 512)                 │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ dense_1 (Dense)                      │ (None, 10)                  │
5,130 │
└──────────────────────────────────────┴─────────────────────────────┴─
─────────────────┘

 Total params: 21,300,554 (81.26 MB)
 Trainable params: 21,283,530 (81.19 MB)

 Non-trainable params: 17,024 (66.50 KB)

Epoch 1/2
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 334s 199ms/step - accuracy: 0.7557 -
loss: 0.7122 - val_accuracy: 0.8728 - val_loss: 0.3428
Epoch 2/2
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 295s 197ms/step - accuracy: 0.8852 -
loss: 0.3134 - val_accuracy: 0.9020 - val_loss: 0.2802
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 69s 46ms/step - accuracy: 0.9111 -
loss: 0.2476
375/375 ━━━━━━━━━━━━━━━━━━━━ 17s 46ms/step - accuracy: 0.9041 - loss:
0.2730
313/313 ━━━━━━━━━━━━━━━━━━━━ 18s 49ms/step - accuracy: 0.9008 - loss:
0.2851

Evaluation Results:
Training Loss: 0.2486  Training Accuracy: 0.9101
Validation Loss: 0.2802  Validation Accuracy: 0.9020
Test Loss: 0.2887  Test Accuracy: 0.9006
```





ResNet - 50

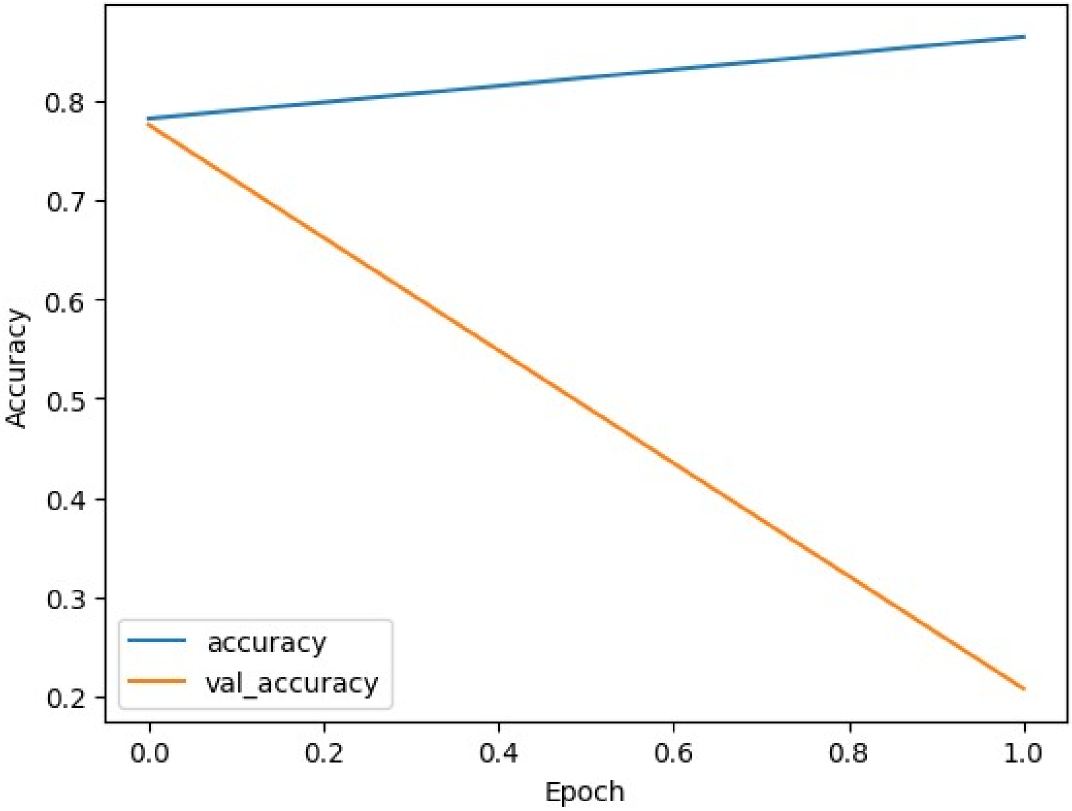
Building the Residual unit for the ResNet50 The 2-layer blocks in Resnet34 was replaced with a 3-layer bottleneck block, forming the Resnet-50 architecture.

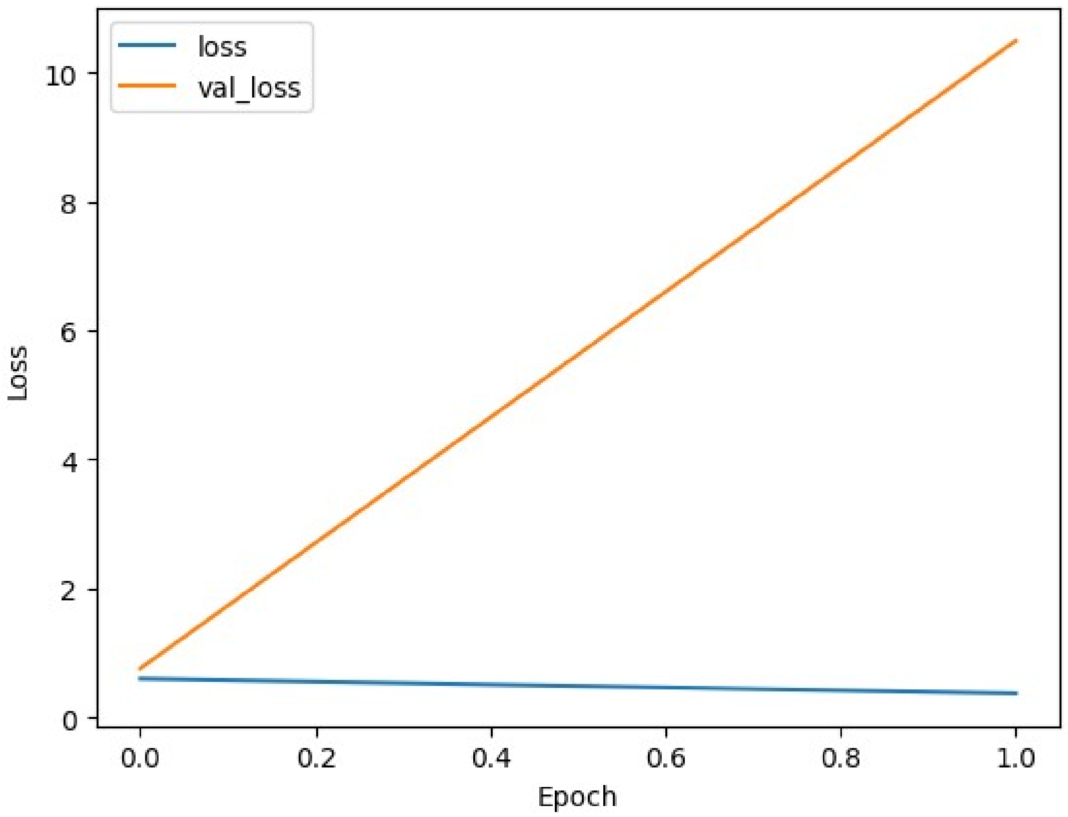
```
import tensorflow
import tensorflow.keras as keras

class ResidualUnit(keras.layers.Layer):
    def __init__(self, filters, strides=1, activation="relu",
**kwargs):
        super().__init__(**kwargs)
        self.activation = keras.activations.get(activation)
        self.main_layers = [
        # The main path of the residual unit consists of two 1x1
convolutions and one 3x3 convolution.
        # The first 1x1 convolution is used to reduce the number of
channels.
        # The 3x3 convolution is used to learn the features.
        # The second 1x1 convolution is used to increase the number of
channels back to the original number.
        keras.layers.Conv2D(filters, 1, strides=strides,
padding="same", use_bias=False),
        keras.layers.BatchNormalization(),
        self.activation,
        keras.layers.Conv2D(filters, 3, strides=1, padding="same",
use_bias=False),
        keras.layers.BatchNormalization(),
        self.activation,
        keras.layers.Conv2D(filters*4, 1, strides=1, padding="same",
use_bias=False),
        keras.layers.BatchNormalization()]
        self.skip_layers = []
        if strides > 1 or filters != kwargs.get("filters", filters*4):
            self.skip_layers = [
                keras.layers.Conv2D(filters*4, 1,
strides=strides,padding="same", use_bias=False),
                keras.layers.BatchNormalization()]



    def call(self, inputs):
        Z = inputs
        skip_Z = inputs
        for layer in self.main_layers:
            Z = layer(Z)
    # Skip connection
        for layer in self.skip_layers:
            skip_Z = layer(skip_Z)
        return self.activation(Z + skip_Z)


from keras import models
from keras import layers
model = keras.models.Sequential()
model.add(keras.layers.Resizing(224, 224,
interpolation="bilinear",input_shape=[28,28,1]))
model.add(layers.Conv2D(64, (7,7), strides=2, padding='same',
use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.ReLU())
model.add(layers.MaxPooling2D((3, 3), strides=2, padding='same'))

Prv = 64

for filters in [64] * 3 + [128] * 4 + [256] * 6 + [512] * 3:
    strides = 1 if filters == Prv else 2
    model.add(ResidualUnit(filters, strides=strides))
    Prv = filters

model.add(layers.GlobalAveragePooling2D())
model.add(layers.Flatten())
model.add(layers.Dense(10, activation='softmax'))

model.summary()


model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
history=model.fit(x_train,y_train, epochs=2, batch_size=32,
validation_data=(x_val,y_val))

#Print the training, validation and test accuracy

train_loss, train_accuracy = model.evaluate(x_train, y_train)
val_loss, val_accuracy = model.evaluate(x_val, y_val)
test_loss, test_accuracy = model.evaluate(x_test, y_test)

print("\nEvaluation Results:")
print("Training Loss: {:.4f}  Training Accuracy:
{:.4f}".format(train_loss, train_accuracy))
print("Validation Loss: {:.4f}  Validation Accuracy:
{:.4f}".format(val_loss, val_accuracy))
print("Test Loss: {:.4f}  Test Accuracy: {:.4f}".format(test_loss,
test_accuracy))


import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
import matplotlib.pyplot as plt
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()


Model: "sequential_2"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
━━━━━━━━━━━━━━━━━┩
│ resizing_2 (Resizing)                │ (None, 224, 224, 1)         │
0 │
```

```
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d_56 (Conv2D)                   │ (None, 112, 112, 64)        │
3,136 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_56               │ (None, 112, 112, 64)        │
256 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ re_lu_2 (ReLU)                       │ (None, 112, 112, 64)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ max_pooling2d_2 (MaxPooling2D)       │ (None, 56, 56, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_24 (ResidualUnit)      │ (None, 56, 56, 256)         │
76,288 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_25 (ResidualUnit)      │ (None, 56, 56, 256)         │
137,728 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_26 (ResidualUnit)      │ (None, 56, 56, 256)         │
137,728 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_27 (ResidualUnit)      │ (None, 28, 28, 512)         │
381,952 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_28 (ResidualUnit)      │ (None, 28, 28, 512)         │
545,792 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_29 (ResidualUnit)      │ (None, 28, 28, 512)         │
545,792 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_30 (ResidualUnit)      │ (None, 28, 28, 512)         │
545,792 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_31 (ResidualUnit)      │ (None, 14, 14, 1024)        │
```

```
1,517,568 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_32 (ResidualUnit)      │ (None, 14, 14, 1024)        │
2,172,928 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_33 (ResidualUnit)      │ (None, 14, 14, 1024)        │
2,172,928 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_34 (ResidualUnit)      │ (None, 14, 14, 1024)        │
2,172,928 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_35 (ResidualUnit)      │ (None, 14, 14, 1024)        │
2,172,928 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_36 (ResidualUnit)      │ (None, 14, 14, 1024)        │
2,172,928 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_37 (ResidualUnit)      │ (None, 7, 7, 2048)          │
6,049,792 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_38 (ResidualUnit)      │ (None, 7, 7, 2048)          │
8,671,232 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ residual_unit_39 (ResidualUnit)      │ (None, 7, 7, 2048)          │
8,671,232 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ global_average_pooling2d_2           │ (None, 2048)                │
0 │
│ (GlobalAveragePooling2D)             │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ flatten_2 (Flatten)                  │ (None, 2048)                │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_2 (Dense)                      │ (None, 10)                  │
20,490 │
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘
```

```
 Total params: 38,169,418 (145.60 MB)

 Trainable params: 38,093,770 (145.32 MB)

 Non-trainable params: 75,648 (295.50 KB)

Epoch 1/2
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 817s 495ms/step - accuracy: 0.7015 -
loss: 0.8511 - val_accuracy: 0.7755 - val_loss: 0.7547
Epoch 2/2
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 734s 489ms/step - accuracy: 0.8535 -
loss: 0.3928 - val_accuracy: 0.2081 - val_loss: 10.4965
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 192s 128ms/step - accuracy: 0.2061 -
loss: 10.6830
375/375 ━━━━━━━━━━━━━━━━━━━━ 48s 128ms/step - accuracy: 0.2066 - loss:
10.4106
313/313 ━━━━━━━━━━━━━━━━━━━━ 48s 141ms/step - accuracy: 0.2062 - loss:
10.4925

Evaluation Results:
Training Loss: 10.6139  Training Accuracy: 0.2040
Validation Loss: 10.4965  Validation Accuracy: 0.2081
Test Loss: 10.5870  Test Accuracy: 0.2009
```




loss
val_loss
10
8
6
4
2
0
Loss
0.0
0.2
0.4
0.6
0.8
1.0
Epoch