```
from google.colab import files
files.upload()
```

Choose Files no files selected Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
{'kaggle.json':
```

```
!mkdir ~/.kaggle/
```

```
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle competitions download -c dogs-vs-cats
```

```
Downloading dogs-vs-cats.zip to /content
 99% 803M/812M [00:09<00:00, 61.4MB/s]
100% 812M/812M [00:09<00:00, 89.3MB/s]
```

```
!unzip -qq dogs-vs-cats.zip
```

```
!unzip -qq train.zip
```

1.Consider the Cats & Dogs example. Start initially with a training sample of 1000, a validation sample of 500, and a test sample of 500 (like in the text). Use any technique to reduce overfitting and improve performance in developing a network that you train from scratch. What performance did you achieve?

Copying pictures to the test, validation, and training sets.

```
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small_1")

def make_subset(subset_name, start_index, end_index):
    for category in ("cat", "dog"):
        dir = new_base_dir / subset_name / category
        os.makedirs(dir)
        fnames = [f"{category}.{i}.jpg" for i in range(start_index, end_index)]
        for fname in fnames:
            shutil.copyfile(src=original_dir / fname,
                            dst=dir / fname)

make_subset("train", start_index=0, end_index=1000)
make_subset("validation", start_index=1000, end_index=1500)
make_subset("test", start_index=1500, end_index=2000)
```

Convolutional neural networks are used.

```
from tensorflow import keras
from tensorflow.keras import layers

inputs = keras.Input(shape=(180, 180, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 180, 180, 3)]     0         
                                                                 
 rescaling (Rescaling)       (None, 180, 180, 3)       0         
                                                                 
 conv2d (Conv2D)             (None, 178, 178, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2  (None, 89, 89, 32)        0         
 D)                                                              
                                                                 
 conv2d_1 (Conv2D)           (None, 87, 87, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 43, 43, 64)        0         
 g2D)                                                            
                                                                 
 conv2d_2 (Conv2D)           (None, 41, 41, 128)       73856     
                                                                 
 max_pooling2d_2 (MaxPoolin  (None, 20, 20, 128)       0         
 g2D)                                                            
                                                                 
 conv2d_3 (Conv2D)           (None, 18, 18, 256)       295168    
                                                                 
 max_pooling2d_3 (MaxPoolin  (None, 9, 9, 256)         0         
 g2D)                                                            
                                                                 
 conv2d_4 (Conv2D)           (None, 7, 7, 256)         590080    
                                                                 
 flatten (Flatten)           (None, 12544)             0         
                                                                 
 dense (Dense)               (None, 1)                 12545     
                                                                 
=================================================================
Total params: 991041 (3.78 MB)
Trainable params: 991041 (3.78 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Given that the model may overfit, regularization techniques are employed at the DATA PREPROCESSING step.

Here, every image is transformed into a tensor.

```
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

```
from tensorflow.keras.utils import image_dataset_from_directory

train_dataset = image_dataset_from_directory(
    new_base_dir / "train",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    new_base_dir / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    new_base_dir / "test",
    image_size=(180, 180),
    batch_size=32)
```

```
Found 2000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

Callback can be either used for saveage the model's weights at the end of every epoch or for early stopping if the model is not improving. Besides that callbacks, such as logging metrics, visualizing the model performance, or scheduling a learning rate changes may be used in the same way.

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch1.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/30
```

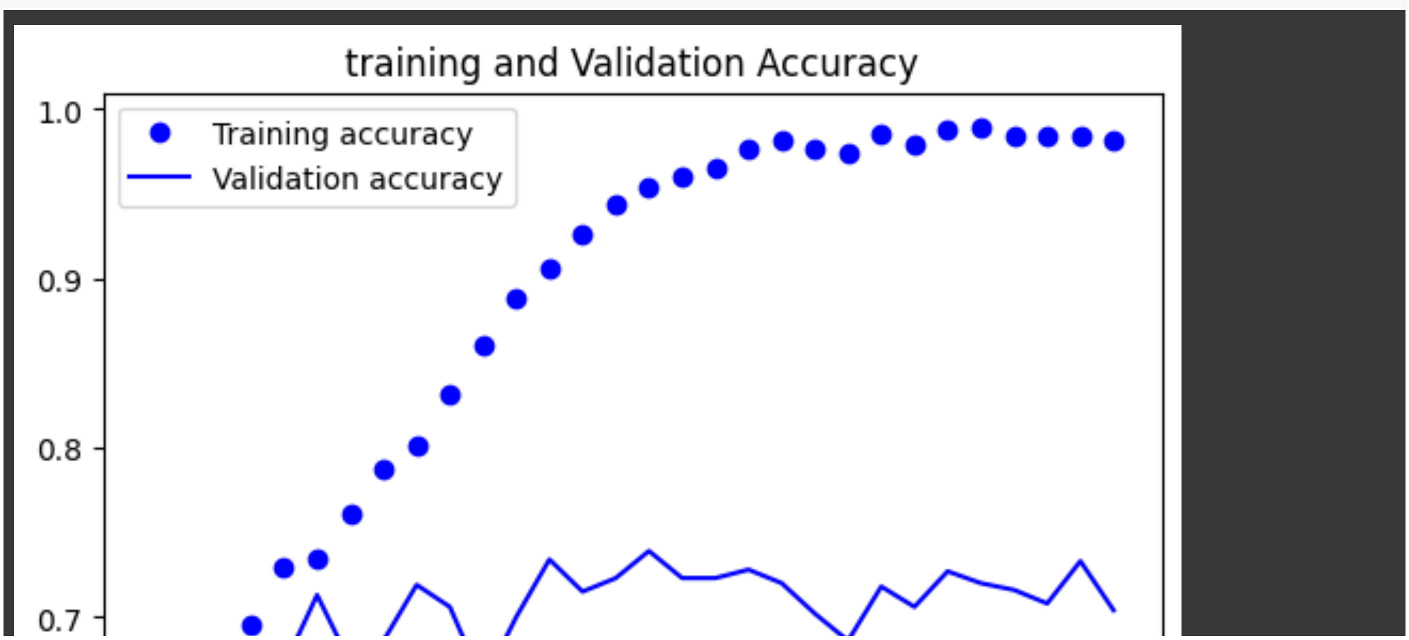
```
63/63 [==============================] - 6s 96ms/step - loss: 0.6868 - accura
Epoch 2/30
63/63 [==============================] - 4s 57ms/step - loss: 0.6487 - accura
Epoch 3/30
63/63 [==============================] - 4s 56ms/step - loss: 0.5969 - accura
Epoch 4/30
63/63 [==============================] - 5s 72ms/step - loss: 0.5681 - accura
Epoch 5/30
63/63 [==============================] - 6s 84ms/step - loss: 0.5399 - accura
Epoch 6/30
63/63 [==============================] - 4s 57ms/step - loss: 0.5165 - accura
Epoch 7/30
63/63 [==============================] - 7s 103ms/step - loss: 0.4856 - accura
Epoch 8/30
63/63 [==============================] - 4s 56ms/step - loss: 0.4560 - accura
Epoch 9/30
63/63 [==============================] - 4s 56ms/step - loss: 0.4115 - accura
Epoch 10/30
63/63 [==============================] - 6s 97ms/step - loss: 0.3636 - accura
Epoch 11/30
63/63 [==============================] - 4s 57ms/step - loss: 0.3278 - accura
Epoch 12/30
63/63 [==============================] - 4s 60ms/step - loss: 0.2721 - accura
Epoch 13/30
63/63 [==============================] - 6s 83ms/step - loss: 0.2251 - accura
Epoch 14/30
63/63 [==============================] - 4s 58ms/step - loss: 0.1930 - accura
Epoch 15/30
63/63 [==============================] - 4s 57ms/step - loss: 0.1472 - accura
Epoch 16/30
63/63 [==============================] - 5s 80ms/step - loss: 0.1295 - accura
Epoch 17/30
63/63 [==============================] - 4s 56ms/step - loss: 0.0976 - accura
Epoch 18/30
63/63 [==============================] - 6s 90ms/step - loss: 0.0945 - accura
Epoch 19/30
63/63 [==============================] - 4s 57ms/step - loss: 0.0674 - accura
Epoch 20/30
63/63 [==============================] - 4s 56ms/step - loss: 0.0546 - accura
Epoch 21/30
63/63 [==============================] - 6s 87ms/step - loss: 0.0719 - accura
Epoch 22/30
63/63 [==============================] - 4s 57ms/step - loss: 0.0767 - accura
Epoch 23/30
63/63 [==============================] - 6s 95ms/step - loss: 0.0494 - accura
Epoch 24/30
63/63 [==============================] - 5s 66ms/step - loss: 0.0561 - accura
Epoch 25/30
63/63 [==============================] - 4s 61ms/step - loss: 0.0406 - accura
Epoch 26/30
```

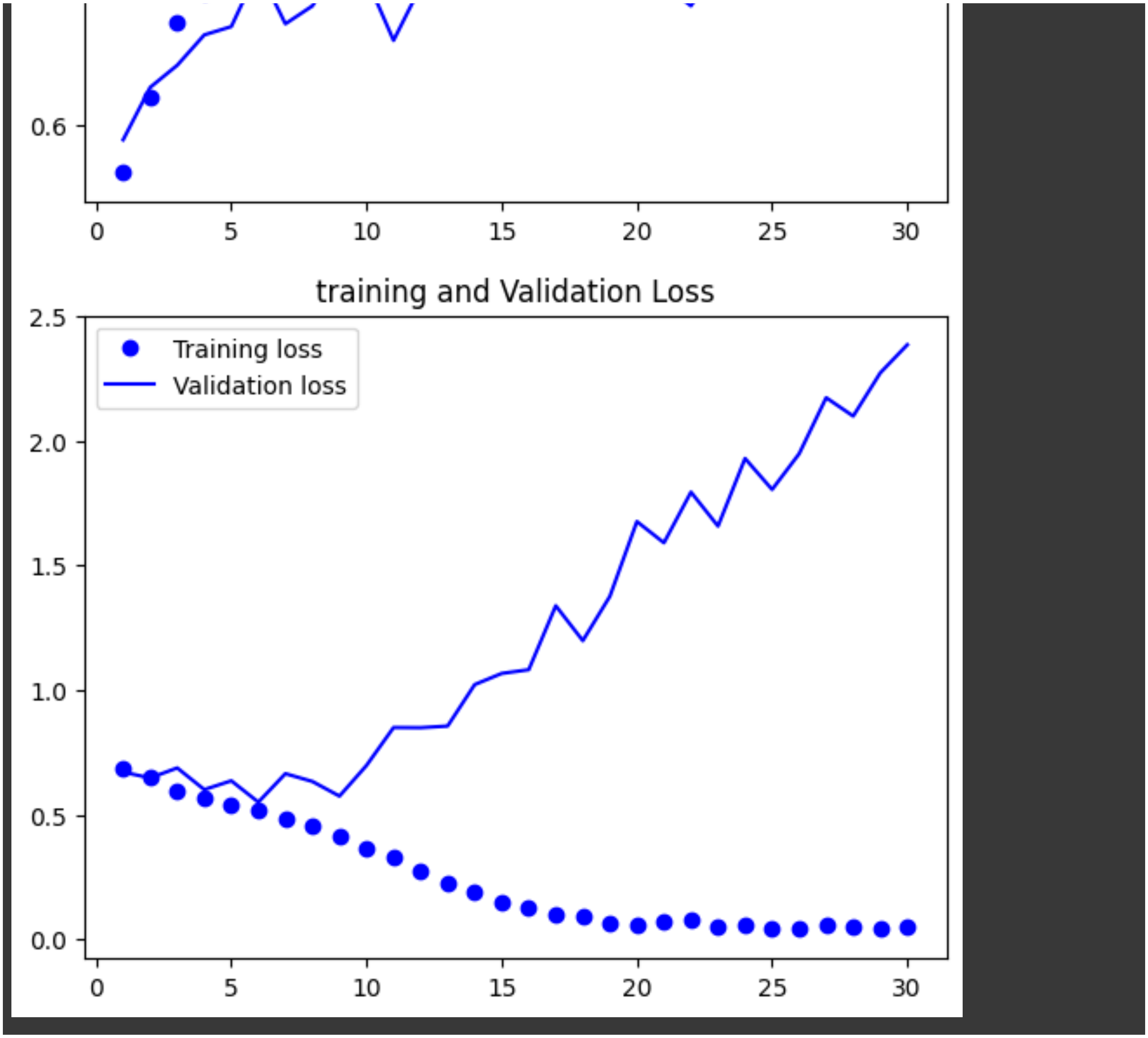
```
63/63 [==============================] - 4s 64ms/step - loss: 0.0462 - accura
Epoch 27/30
63/63 [==============================] - 7s 103ms/step - loss: 0.0544 - accur
Epoch 28/30
63/63 [==============================] - 4s 60ms/step - loss: 0.0533 - accura
Epoch 29/30
63/63 [==============================] - 6s 95ms/step - loss: 0.0435 - accura
Epoch 30/30
```

Accuracy appears to be rising with the number of epochs.

Accuracy=71.3 Val_acc=70.4 test_acc=71.3

```
import matplotlib.pyplot as plt
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("training and Validation Accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("training and Validation Loss")
plt.legend()
plt.show()
```






```
test_model = keras.models.load_model("convnet_from_scratch1.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 28ms/step - loss: 0.5793 - accura
Test accuracy: 0.713
```

Test accuracy with no data augmentation=71.3

Data Augmentation

Data augmentation acts like a master artist adding new custom made artwork to the training dataset by artfully modifying the original data. Apart from assisting in curbing overfitting, this aesthetic touch also leads to the development of a broader capability of recognizing more generalized cases by the model.

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation1.keras",
```

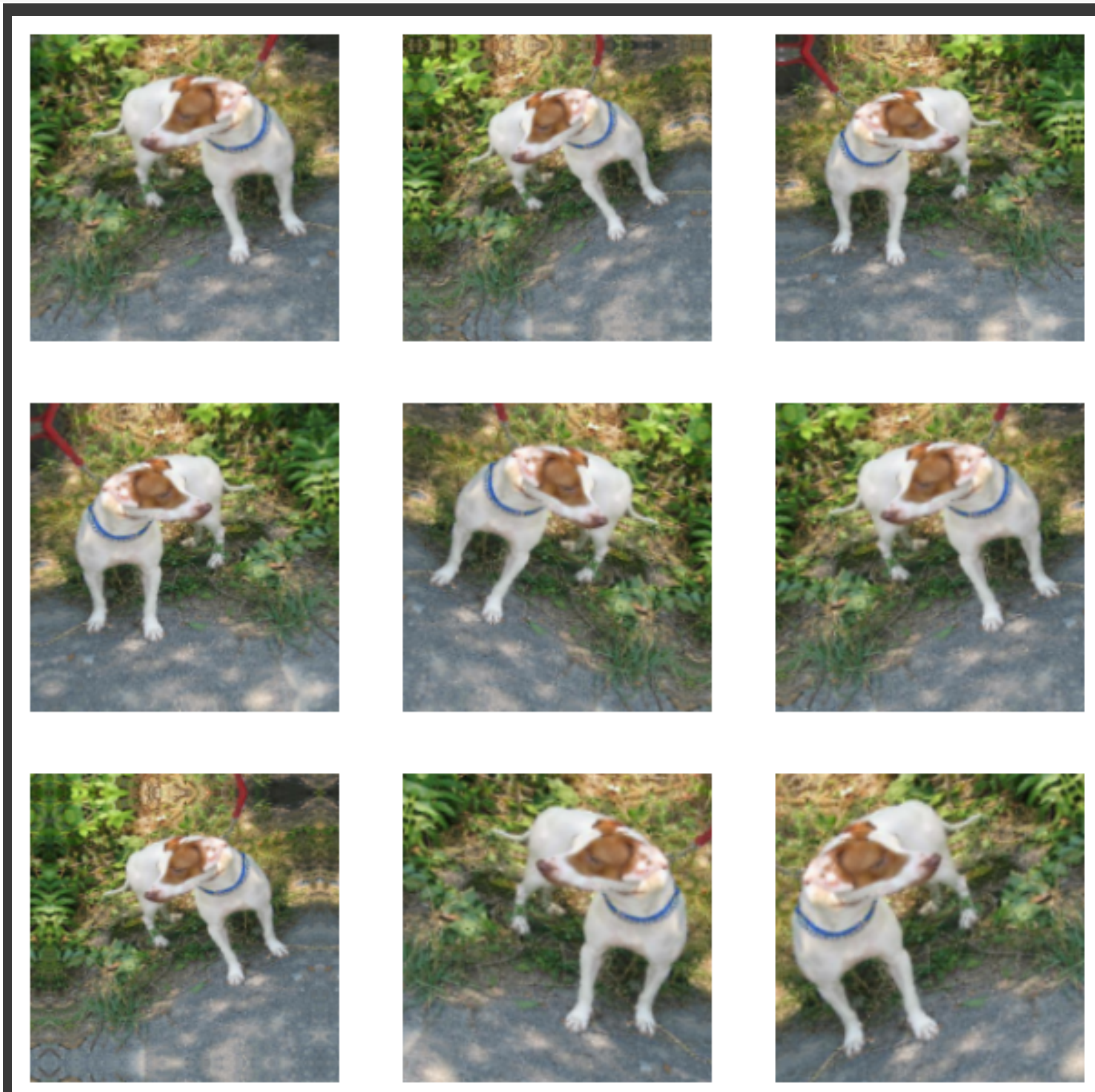
```
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=50,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/50
63/63 [==============================] - 9s 95ms/step - loss: 0.6988 - accurac
Epoch 2/50
63/63 [==============================] - 5s 75ms/step - loss: 0.6943 - accurac
Epoch 3/50
63/63 [==============================] - 6s 85ms/step - loss: 0.6910 - accurac
Epoch 4/50
63/63 [==============================] - 6s 83ms/step - loss: 0.6906 - accurac
Epoch 5/50
63/63 [==============================] - 4s 62ms/step - loss: 0.6798 - accurac
Epoch 6/50
63/63 [==============================] - 6s 93ms/step - loss: 0.6610 - accurac
Epoch 7/50
63/63 [==============================] - 4s 59ms/step - loss: 0.6469 - accurac
Epoch 8/50
63/63 [==============================] - 4s 60ms/step - loss: 0.6402 - accurac
Epoch 9/50
63/63 [==============================] - 8s 114ms/step - loss: 0.6243 - accura
Epoch 10/50
63/63 [==============================] - 5s 67ms/step - loss: 0.6165 - accurac
Epoch 11/50
63/63 [==============================] - 6s 86ms/step - loss: 0.5941 - accurac
Epoch 12/50
63/63 [==============================] - 4s 60ms/step - loss: 0.5913 - accurac
Epoch 13/50
63/63 [==============================] - 6s 86ms/step - loss: 0.5724 - accurac
Epoch 14/50
63/63 [==============================] - 6s 85ms/step - loss: 0.5659 - accurac
Epoch 15/50
63/63 [==============================] - 4s 60ms/step - loss: 0.5640 - accurac
Epoch 16/50
63/63 [==============================] - 4s 62ms/step - loss: 0.5350 - accurac
Epoch 17/50
63/63 [==============================] - 6s 90ms/step - loss: 0.5368 - accurac
Epoch 18/50
63/63 [==============================] - 4s 58ms/step - loss: 0.5253 - accurac
Epoch 19/50
63/63 [==============================] - 4s 62ms/step - loss: 0.5116 - accurac
Epoch 20/50
```

```
63/63 [==============================] - 6s 90ms/step - loss: 0.5123 - accura
Epoch 21/50
63/63 [==============================] - 4s 59ms/step - loss: 0.4917 - accura
Epoch 22/50
63/63 [==============================] - 7s 110ms/step - loss: 0.4862 - accura
Epoch 23/50
63/63 [==============================] - 4s 58ms/step - loss: 0.4734 - accura
Epoch 24/50
63/63 [==============================] - 5s 71ms/step - loss: 0.4826 - accura
Epoch 25/50
63/63 [==============================] - 6s 91ms/step - loss: 0.4590 - accura
Epoch 26/50
63/63 [==============================] - 4s 58ms/step - loss: 0.4477 - accura
Epoch 27/50
63/63 [==============================] - 7s 101ms/step - loss: 0.4377 - accura
Epoch 28/50
63/63 [==============================] - 4s 62ms/step - loss: 0.4309 - accura
Epoch 29/50
63/63 [==============================] - 4s 59ms/step - loss: 0.4420 - accura
Epoch 30/50
```

Displaying some randomly augmented training images:

```
plt.figure(figsize=(7.5,7.5 ))
for images, _ in train_dataset.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_images[0].numpy().astype("uint8"))
        plt.axis("off")
```



2. Increase your training sample size. You may pick any amount. Keep the validation and test samples the same as above

Tried to raise the 1000–1500 training sample size.

```
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small_2")

def make_subset(subset_name, start_index, end_index):
    for category in ("cat", "dog"):
        dir = new_base_dir / subset_name / category
        os.makedirs(dir, exist_ok=True)
        fnames = [f"{category}.{i}.jpg" for i in range(start_index, end_index)]
        for fname in fnames:
            shutil.copyfile(src=original_dir / fname,
                            dst=dir / fname)

make_subset("train", start_index=0, end_index=1500)
make_subset("validation", start_index=1500, end_index=2000)
make_subset("test", start_index=2000, end_index=2500)
```

```
from tensorflow.keras.utils import image_dataset_from_directory

train_dataset = image_dataset_from_directory(
    new_base_dir / "train",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    new_base_dir / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    new_base_dir / "test",
    image_size=(180, 180),
    batch_size=32)
```

```
Found 3000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

```
inputs = keras.Input(shape=(180, 180, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch2.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=70,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/70
94/94 [==============================] - 9s 78ms/step - loss: 0.6923 - accura
Epoch 2/70
94/94 [==============================] - 7s 69ms/step - loss: 0.6699 - accura
Epoch 3/70
94/94 [==============================] - 5s 52ms/step - loss: 0.6373 - accura
Epoch 4/70
94/94 [==============================] - 8s 78ms/step - loss: 0.5955 - accura
Epoch 5/70
94/94 [==============================] - 5s 52ms/step - loss: 0.5593 - accura
Epoch 6/70
94/94 [==============================] - 7s 71ms/step - loss: 0.5061 - accura
Epoch 7/70
```

```
94/94 [==============================] - 6s 64ms/step - loss: 0.4537 - accura
Epoch 8/70
94/94 [==============================] - 5s 51ms/step - loss: 0.4003 - accura
Epoch 9/70
94/94 [==============================] - 7s 75ms/step - loss: 0.3641 - accura
Epoch 10/70
94/94 [==============================] - 6s 61ms/step - loss: 0.2999 - accura
Epoch 11/70
94/94 [==============================] - 6s 57ms/step - loss: 0.2355 - accura
Epoch 12/70
94/94 [==============================] - 9s 93ms/step - loss: 0.1936 - accura
Epoch 13/70
94/94 [==============================] - 5s 54ms/step - loss: 0.1570 - accura
Epoch 14/70
94/94 [==============================] - 7s 70ms/step - loss: 0.1067 - accura
Epoch 15/70
94/94 [==============================] - 5s 51ms/step - loss: 0.0928 - accura
Epoch 16/70
94/94 [==============================] - 7s 72ms/step - loss: 0.0842 - accura
Epoch 17/70
94/94 [==============================] - 7s 64ms/step - loss: 0.0665 - accura
Epoch 18/70
94/94 [==============================] - 7s 73ms/step - loss: 0.0608 - accura
Epoch 19/70
94/94 [==============================] - 6s 63ms/step - loss: 0.0555 - accura
Epoch 20/70
94/94 [==============================] - 5s 51ms/step - loss: 0.0509 - accura
Epoch 21/70
94/94 [==============================] - 8s 83ms/step - loss: 0.0356 - accura
Epoch 22/70
94/94 [==============================] - 5s 52ms/step - loss: 0.0377 - accura
Epoch 23/70
94/94 [==============================] - 9s 92ms/step - loss: 0.0521 - accura
Epoch 24/70
94/94 [==============================] - 5s 51ms/step - loss: 0.0394 - accura
Epoch 25/70
94/94 [==============================] - 6s 57ms/step - loss: 0.0336 - accura
Epoch 26/70
94/94 [==============================] - 8s 77ms/step - loss: 0.0472 - accura
Epoch 27/70
94/94 [==============================] - 5s 51ms/step - loss: 0.0496 - accura
Epoch 28/70
94/94 [==============================] - 7s 69ms/step - loss: 0.0310 - accura
Epoch 29/70
94/94 [==============================] - 7s 70ms/step - loss: 0.0512 - accura
Epoch 30/70
```

```
test_model = keras.models.load_model(
    "convnet_from_scratch2.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 2s 33ms/step - loss: 0.5028 - accura
Test accuracy: 0.758
```

Accuracy=75.8 val_acc=74.6 test_acc=75.8

Applying data augmentation

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.compile(loss="binary_crossentropy",
              optimizer="adam",
              metrics=["accuracy"])
```

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation2.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=80,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/80
94/94 [==============================] - 6s 57ms/step - loss: 0.6896 - accura
Epoch 2/80
94/94 [==============================] - 8s 81ms/step - loss: 0.6922 - accura
Epoch 3/80
94/94 [==============================] - 6s 56ms/step - loss: 0.6827 - accura
Epoch 4/80
94/94 [==============================] - 9s 93ms/step - loss: 0.6571 - accura
Epoch 5/80
94/94 [==============================] - 5s 55ms/step - loss: 0.6494 - accura
Epoch 6/80
```

```
94/94 [==============================] - 7s 68ms/step - loss: 0.6400 - accura
Epoch 7/80
94/94 [==============================] - 5s 53ms/step - loss: 0.6266 - accura
Epoch 8/80
94/94 [==============================] - 7s 72ms/step - loss: 0.5966 - accura
Epoch 9/80
94/94 [==============================] - 5s 53ms/step - loss: 0.5968 - accura
Epoch 10/80
94/94 [==============================] - 7s 69ms/step - loss: 0.5732 - accura
Epoch 11/80
94/94 [==============================] - 7s 72ms/step - loss: 0.5598 - accura
Epoch 12/80
94/94 [==============================] - 6s 63ms/step - loss: 0.5337 - accura
Epoch 13/80
94/94 [==============================] - 5s 53ms/step - loss: 0.5350 - accura
Epoch 14/80
94/94 [==============================] - 6s 56ms/step - loss: 0.4995 - accura
Epoch 15/80
94/94 [==============================] - 5s 52ms/step - loss: 0.5053 - accura
Epoch 16/80
94/94 [==============================] - 8s 85ms/step - loss: 0.4901 - accura
Epoch 17/80
94/94 [==============================] - 5s 52ms/step - loss: 0.4780 - accura
Epoch 18/80
94/94 [==============================] - 8s 78ms/step - loss: 0.4719 - accura
Epoch 19/80
94/94 [==============================] - 5s 56ms/step - loss: 0.4754 - accura
Epoch 20/80
94/94 [==============================] - 7s 72ms/step - loss: 0.4445 - accura
Epoch 21/80
94/94 [==============================] - 6s 63ms/step - loss: 0.4477 - accura
Epoch 22/80
94/94 [==============================] - 7s 68ms/step - loss: 0.4319 - accura
Epoch 23/80
94/94 [==============================] - 5s 53ms/step - loss: 0.4268 - accura
Epoch 24/80
94/94 [==============================] - 6s 64ms/step - loss: 0.4227 - accura
Epoch 25/80
94/94 [==============================] - 5s 52ms/step - loss: 0.4176 - accura
Epoch 26/80
94/94 [==============================] - 7s 72ms/step - loss: 0.3990 - accura
Epoch 27/80
94/94 [==============================] - 5s 53ms/step - loss: 0.3953 - accura
Epoch 28/80
94/94 [==============================] - 7s 71ms/step - loss: 0.3984 - accura
Epoch 29/80
94/94 [==============================] - 7s 68ms/step - loss: 0.3863 - accura
Epoch 30/80
```

Accuracy=93.8 val_acc=85.2 test_acc=84.8

```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation2.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 2s 36ms/step - loss: 0.3820 - accura
Test accuracy: 0.848
```

3. The objective is to find the ideal training sample size to get best prediction

The sizes of the training, validation, and test sets were established at 1500, 1000, and 500, respectively.

```
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small_3")

def make_subset(subset_name, start_index, end_index):
    for category in ("cat", "dog"):
        dir = new_base_dir / subset_name / category
        os.makedirs(dir, exist_ok=True)
        fnames = [f"{category}.{i}.jpg" for i in range(start_index, end_index)]
        for fname in fnames:
            shutil.copyfile(src=original_dir / fname,
                            dst=dir / fname)

make_subset("train", start_index=0, end_index=1500)
make_subset("validation", start_index=1500, end_index=2500)
make_subset("test", start_index=2500, end_index=3000)
```

```
from tensorflow.keras.utils import image_dataset_from_directory

train_dataset = image_dataset_from_directory(
    new_base_dir / "train",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    new_base_dir / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    new_base_dir / "test",
    image_size=(180, 180),
    batch_size=32)
```

```
Found 3000 files belonging to 2 classes.
Found 2000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

```
inputs = keras.Input(shape=(180, 180, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch3.keras",
        save_best_only=True,
        monitor="val_loss")
```

```
]
history = model.fit(
    train_dataset,
    epochs=90,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/90
94/94 [==============================] - 7s 72ms/step - loss: 0.6906 - accura
Epoch 2/90
94/94 [==============================] - 6s 61ms/step - loss: 0.6559 - accura
Epoch 3/90
94/94 [==============================] - 8s 78ms/step - loss: 0.6158 - accura
Epoch 4/90
94/94 [==============================] - 7s 67ms/step - loss: 0.5944 - accura
Epoch 5/90
94/94 [==============================] - 8s 84ms/step - loss: 0.5553 - accura
Epoch 6/90
94/94 [==============================] - 6s 61ms/step - loss: 0.5029 - accura
Epoch 7/90
94/94 [==============================] - 8s 80ms/step - loss: 0.4587 - accura
Epoch 8/90
94/94 [==============================] - 6s 60ms/step - loss: 0.4258 - accura
Epoch 9/90
94/94 [==============================] - 9s 98ms/step - loss: 0.3608 - accura
Epoch 10/90
94/94 [==============================] - 7s 67ms/step - loss: 0.3092 - accura
Epoch 11/90
94/94 [==============================] - 8s 86ms/step - loss: 0.2478 - accura
Epoch 12/90
94/94 [==============================] - 6s 61ms/step - loss: 0.2054 - accura
Epoch 13/90
94/94 [==============================] - 8s 83ms/step - loss: 0.1628 - accura
Epoch 14/90
94/94 [==============================] - 7s 69ms/step - loss: 0.1234 - accura
Epoch 15/90
94/94 [==============================] - 6s 65ms/step - loss: 0.0944 - accura
Epoch 16/90
94/94 [==============================] - 9s 89ms/step - loss: 0.0906 - accura
Epoch 17/90
94/94 [==============================] - 7s 67ms/step - loss: 0.0660 - accura
Epoch 18/90
94/94 [==============================] - 9s 94ms/step - loss: 0.0746 - accura
Epoch 19/90
94/94 [==============================] - 6s 60ms/step - loss: 0.0463 - accura
Epoch 20/90
94/94 [==============================] - 9s 89ms/step - loss: 0.0588 - accura
Epoch 21/90
```

```
94/94 [==============================] - 7s 67ms/step - loss: 0.0366 - accura
Epoch 22/90
94/94 [==============================] - 7s 67ms/step - loss: 0.0454 - accura
Epoch 23/90
94/94 [==============================] - 9s 93ms/step - loss: 0.0344 - accura
Epoch 24/90
94/94 [==============================] - 6s 61ms/step - loss: 0.0464 - accura
Epoch 25/90
94/94 [==============================] - 7s 74ms/step - loss: 0.0591 - accura
Epoch 26/90
94/94 [==============================] - 8s 79ms/step - loss: 0.0348 - accura
Epoch 27/90
94/94 [==============================] - 9s 96ms/step - loss: 0.0388 - accura
Epoch 28/90
94/94 [==============================] - 6s 60ms/step - loss: 0.0358 - accura
Epoch 29/90
94/94 [==============================] - 8s 78ms/step - loss: 0.0654 - accura
Epoch 30/90
```

```
test_model = keras.models.load_model(
    "convnet_from_scratch3.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 28ms/step - loss: 0.5366 - accura
Test accuracy: 0.752
```

Accuracy=99.3 val_Acc=75.0 test_Acc=75.2

Utilizing Augmented Data

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.compile(loss="binary_crossentropy",
              optimizer="adam",
              metrics=["accuracy"])
```

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation3.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=100,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/100
94/94 [==============================] - 13s 107ms/step - loss: 0.6933 - accu
Epoch 2/100
94/94 [==============================] - 9s 94ms/step - loss: 0.6943 - accura
Epoch 3/100
94/94 [==============================] - 6s 62ms/step - loss: 0.6941 - accura
Epoch 4/100
94/94 [==============================] - 7s 71ms/step - loss: 0.6935 - accura
Epoch 5/100
94/94 [==============================] - 9s 92ms/step - loss: 0.6934 - accura
Epoch 6/100
94/94 [==============================] - 6s 64ms/step - loss: 0.6933 - accura
```

```
Epoch 7/100
94/94 [==============================] - 10s 104ms/step - loss: 0.6934 - accu
Epoch 8/100
94/94 [==============================] - 6s 61ms/step - loss: 0.6934 - accura
Epoch 9/100
94/94 [==============================] - 9s 89ms/step - loss: 0.6958 - accura
Epoch 10/100
94/94 [==============================] - 6s 62ms/step - loss: 0.6934 - accura
Epoch 11/100
94/94 [==============================] - 7s 69ms/step - loss: 0.6921 - accura
Epoch 12/100
94/94 [==============================] - 10s 109ms/step - loss: 0.6945 - accu
Epoch 13/100
94/94 [==============================] - 6s 62ms/step - loss: 0.6933 - accura
Epoch 14/100
94/94 [==============================] - 8s 84ms/step - loss: 0.6933 - accura
Epoch 15/100
94/94 [==============================] - 9s 96ms/step - loss: 0.6932 - accura
Epoch 16/100
94/94 [==============================] - 7s 68ms/step - loss: 0.6927 - accura
Epoch 17/100
94/94 [==============================] - 9s 96ms/step - loss: 0.6932 - accura
Epoch 18/100
94/94 [==============================] - 7s 66ms/step - loss: 0.6934 - accura
Epoch 19/100
94/94 [==============================] - 7s 71ms/step - loss: 0.6931 - accura
Epoch 20/100
94/94 [==============================] - 9s 86ms/step - loss: 0.6938 - accura
Epoch 21/100
94/94 [==============================] - 6s 61ms/step - loss: 0.6929 - accura
Epoch 22/100
94/94 [==============================] - 12s 122ms/step - loss: 0.6932 - accu
Epoch 23/100
94/94 [==============================] - 6s 61ms/step - loss: 0.6930 - accura
Epoch 24/100
94/94 [==============================] - 7s 65ms/step - loss: 0.6930 - accura
Epoch 25/100
94/94 [==============================] - 9s 96ms/step - loss: 0.6929 - accura
Epoch 26/100
94/94 [==============================] - 6s 62ms/step - loss: 0.6942 - accura
Epoch 27/100
94/94 [==============================] - 8s 81ms/step - loss: 0.6933 - accura
Epoch 28/100
94/94 [==============================] - 8s 75ms/step - loss: 0.6926 - accura
Epoch 29/100
94/94 [==============================] - 7s 68ms/step - loss: 0.6936 - accura
Epoch 30/100
```

```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation3.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 28ms/step - loss: 0.6919 - accura
Test accuracy: 0.524
```

Accuracy=48.9 val_acc=50.0 test_acc=52.4

4. Now using a pretrained network.

This pre-trained network is modeled by VGG16.

Creating the VGG16 convolutional basis is the first step in feature extraction.

```
conv_base = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False,
    input_shape=(180, 180, 3))
conv_base.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applica
58889256/58889256 [==============================] - 0s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, 180, 180, 3)]     0

 block1_conv1 (Conv2D)       (None, 180, 180, 64)      1792

 block1_conv2 (Conv2D)       (None, 180, 180, 64)      36928

 block1_pool (MaxPooling2D)  (None, 90, 90, 64)        0

 block2_conv1 (Conv2D)       (None, 90, 90, 128)       73856

 block2_conv2 (Conv2D)       (None, 90, 90, 128)       147584

 block2_pool (MaxPooling2D)  (None, 45, 45, 128)       0

 block3_conv1 (Conv2D)       (None, 45, 45, 256)       295168

 block3_conv2 (Conv2D)       (None, 45, 45, 256)       590080

 block3_conv3 (Conv2D)       (None, 45, 45, 256)       590080

 block3_pool (MaxPooling2D)  (None, 22, 22, 256)       0

 block4_conv1 (Conv2D)       (None, 22, 22, 512)       1180160

 block4_conv2 (Conv2D)       (None, 22, 22, 512)       2359808

 block4_conv3 (Conv2D)       (None, 22, 22, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 11, 11, 512)       0

 block5_conv1 (Conv2D)       (None, 11, 11, 512)       2359808

 block5_conv2 (Conv2D)       (None, 11, 11, 512)       2359808

 block5_conv3 (Conv2D)       (None, 11, 11, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 5, 5, 512)         0

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Extraction of features and associated labels is known as feature extraction.

```
import numpy as np

def get_features_and_labels(dataset):
    all_features = []
    all_labels = []
    for images, labels in dataset:
        preprocessed_images = keras.applications.vgg16.preprocess_input(images)
        features = conv_base.predict(preprocessed_images)
        all_features.append(features)
        all_labels.append(labels)
    return np.concatenate(all_features), np.concatenate(all_labels)

train_features, train_labels =  get_features_and_labels(train_dataset)
val_features, val_labels =  get_features_and_labels(validation_dataset)
test_features, test_labels =  get_features_and_labels(test_dataset)

train_features.shape
```

```
1/1 [==============================] - 5s 5s/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 33ms/step
```

```
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
```

The densely connected classifier is defined and trained throughout the feature extraction process.

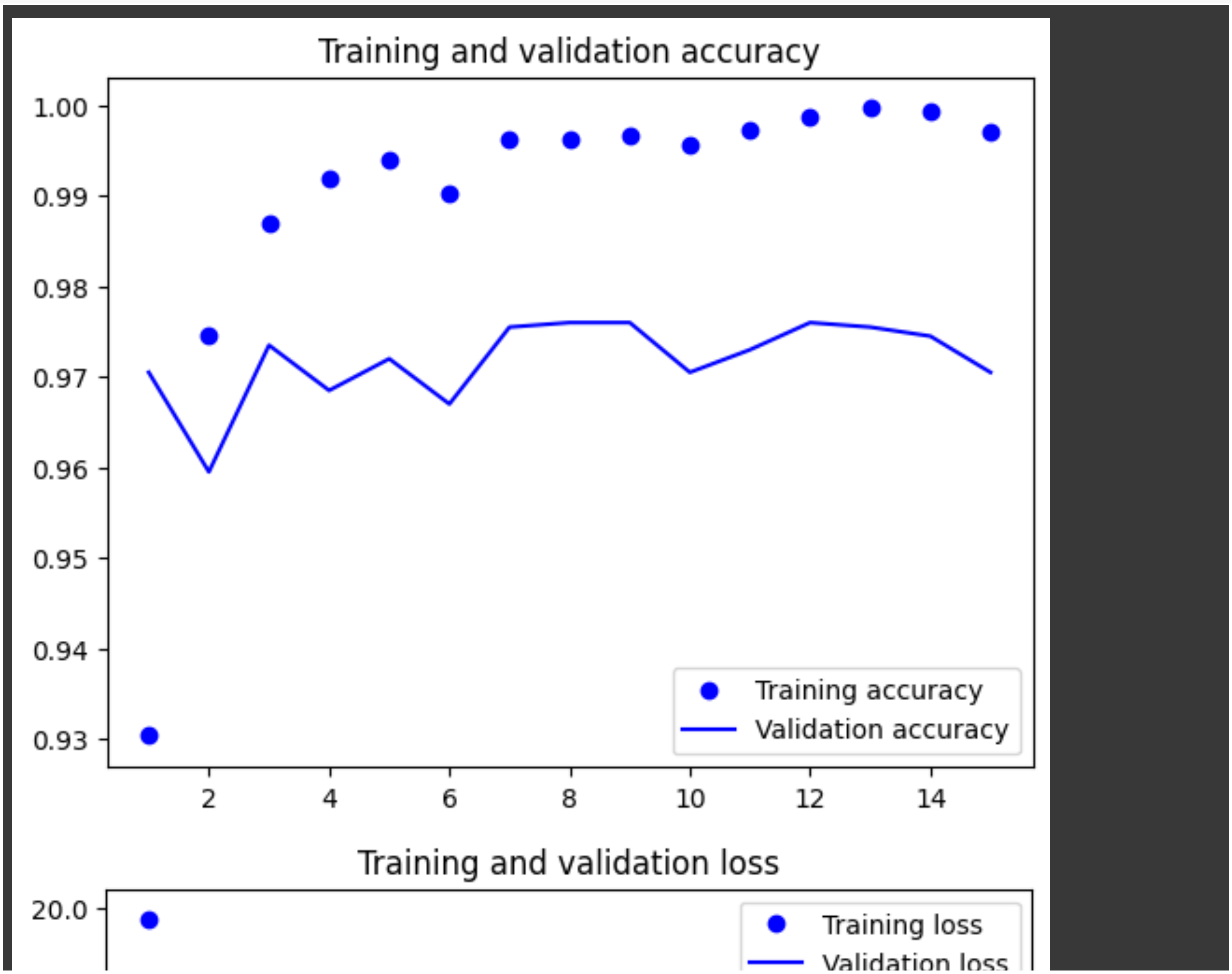
```
inputs = keras.Input(shape=(5, 5, 512))
x = layers.Flatten()(inputs)
x = layers.Dense(256)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

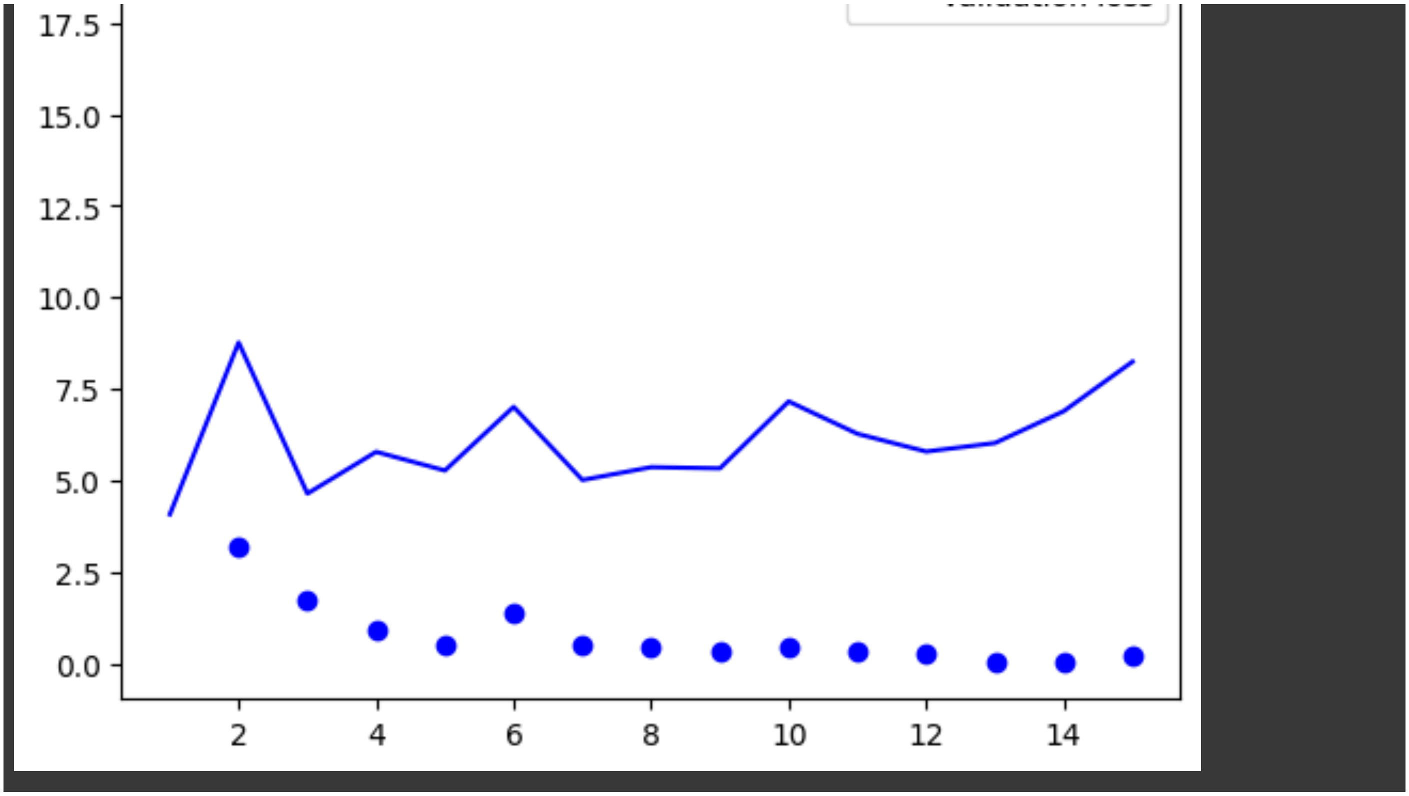
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
      filepath="feature_extractionPT1.keras",
      save_best_only=True,
      monitor="val_loss")
]
history = model.fit(
    train_features, train_labels,
    epochs=15,
    validation_data=(val_features, val_labels),
    callbacks=callbacks)
```

```
Epoch 1/15
94/94 [==============================] - 2s 10ms/step - loss: 19.6264 - accura
Epoch 2/15
94/94 [==============================] - 0s 5ms/step - loss: 3.1697 - accuracy
Epoch 3/15
94/94 [==============================] - 0s 5ms/step - loss: 1.6974 - accuracy
Epoch 4/15
94/94 [==============================] - 0s 5ms/step - loss: 0.9081 - accuracy
Epoch 5/15
94/94 [==============================] - 0s 5ms/step - loss: 0.4778 - accuracy
Epoch 6/15
94/94 [==============================] - 1s 8ms/step - loss: 1.3842 - accuracy
Epoch 7/15
94/94 [==============================] - 1s 8ms/step - loss: 0.5023 - accuracy
Epoch 8/15
94/94 [==============================] - 1s 8ms/step - loss: 0.4250 - accuracy
Epoch 9/15
94/94 [==============================] - 1s 8ms/step - loss: 0.3422 - accuracy
Epoch 10/15
94/94 [==============================] - 1s 8ms/step - loss: 0.4261 - accuracy
Epoch 11/15
94/94 [==============================] - 1s 8ms/step - loss: 0.3498 - accuracy
Epoch 12/15
94/94 [==============================] - 1s 8ms/step - loss: 0.2699 - accuracy
Epoch 13/15
94/94 [==============================] - 1s 9ms/step - loss: 0.0019 - accuracy
Epoch 14/15
94/94 [==============================] - 1s 7ms/step - loss: 0.0149 - accuracy
Epoch 15/15
94/94 [==============================] - 1s 5ms/step - loss: 0.2001 - accuracy
```

accuracy=99.7 val_acc=97.5

```
import matplotlib.pyplot as plt
acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, "bo", label="Training accuracy")
plt.plot(epochs, val_acc, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```






```
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False)
conv_base.trainable = False


conv_base.trainable = True
print("This is the number of trainable weights "
      "before freezing the conv base:", len(conv_base.trainable_weights))


conv_base.trainable = False
print("This is the number of trainable weights "
      "after freezing the conv base:", len(conv_base.trainable_weights))
```

```
This is the number of trainable weights before freezing the conv base: 26
This is the number of trainable weights after freezing the conv base: 0
```

Utilizing data augmentation for feature extraction

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)

inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = keras.applications.vgg16.preprocess_input(x)
x = conv_base(x)
x = layers.Flatten()(x)
x = layers.Dense(256)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

```
import tensorflow as tf
from tensorflow import keras
```

```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/loc
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.1
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/pyth
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.1
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/lo
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.1
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/pytho
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/loca
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.1
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/di
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.1
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/l
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/pyth
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyth
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/di
```

```
import tensorflow as tf
from tensorflow import keras

callbacks = [
```

```
    keras.callbacks.ModelCheckpoint(
        filepath="feature_extraction_with_data_augmentationPT2.keras",
        save_best_only=True,
        monitor="val_loss",
        save_weights_only=True)
]
history = model.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/30
94/94 [==============================] - 16s 171ms/step - loss: 1.3165 - accu
Epoch 2/30
94/94 [==============================] - 17s 174ms/step - loss: 1.7702 - accu
Epoch 3/30
94/94 [==============================] - 16s 171ms/step - loss: 1.4329 - accu
Epoch 4/30
94/94 [==============================] - 16s 166ms/step - loss: 1.3123 - accu
Epoch 5/30
94/94 [==============================] - 16s 167ms/step - loss: 1.0921 - accu
Epoch 6/30
94/94 [==============================] - 20s 213ms/step - loss: 0.9200 - accu
Epoch 7/30
94/94 [==============================] - 16s 166ms/step - loss: 0.9962 - accu
Epoch 8/30
94/94 [==============================] - 20s 210ms/step - loss: 0.8914 - accu
Epoch 9/30
94/94 [==============================] - 16s 167ms/step - loss: 0.6181 - accu
Epoch 10/30
94/94 [==============================] - 16s 171ms/step - loss: 0.7317 - accu
Epoch 11/30
94/94 [==============================] - 16s 168ms/step - loss: 0.6439 - accu
Epoch 12/30
94/94 [==============================] - 16s 169ms/step - loss: 0.8221 - accu
Epoch 13/30
94/94 [==============================] - 16s 165ms/step - loss: 0.7171 - accu
Epoch 14/30
94/94 [==============================] - 16s 167ms/step - loss: 0.5458 - accu
Epoch 15/30
94/94 [==============================] - 16s 169ms/step - loss: 0.6724 - accu
Epoch 16/30
94/94 [==============================] - 16s 167ms/step - loss: 0.5941 - accu
Epoch 17/30
94/94 [==============================] - 16s 167ms/step - loss: 0.5272 - accu
Epoch 18/30
94/94 [==============================] - 16s 166ms/step - loss: 0.4459 - accu
Epoch 19/30
```

```
94/94 [==============================] - 16s 165ms/step - loss: 0.4504 - accu
Epoch 20/30
94/94 [==============================] - 16s 168ms/step - loss: 0.6865 - accu
Epoch 21/30
94/94 [==============================] - 20s 210ms/step - loss: 0.5540 - accu
Epoch 22/30
94/94 [==============================] - 20s 207ms/step - loss: 0.6136 - accu
Epoch 23/30
94/94 [==============================] - 16s 166ms/step - loss: 0.5023 - accu
Epoch 24/30
94/94 [==============================] - 20s 210ms/step - loss: 0.4436 - accu
Epoch 25/30
94/94 [==============================] - 16s 167ms/step - loss: 0.5005 - accu
Epoch 26/30
94/94 [==============================] - 16s 166ms/step - loss: 0.4109 - accu
Epoch 27/30
94/94 [==============================] - 20s 209ms/step - loss: 0.3969 - accu
Epoch 28/30
94/94 [==============================] - 16s 166ms/step - loss: 0.5013 - accu
Epoch 29/30
94/94 [==============================] - 20s 208ms/step - loss: 0.5413 - accu
Epoch 30/30
94/94 [==============================] - 16s 167ms/step - loss: 0.6312 - accu
```

```
test_model = keras.models.load_model(
    "feature_extraction_with_data_augmentationPT2.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

32/32 [==============================] - 3s 89ms/step - loss: 4.2593 - accuracy: 0.9690

Test accuracy: 0.969 accuracy=96.9% val_Acc=97.4% test_acc=97.4%