aml-assignment-3

March 25, 2025

[2]:
```
# Linking drive to the current collab file
from google.colab import drive
drive. mount('/content/drive')
```

```
Mounted at /content/drive
```

[4]:
```
# # Installing the 'patool' package to extract files
pip install patool
```

```
Collecting patool
  Downloading patool-4.0.0-py2.py3-none-any.whl.metadata (4.5 kB)
Downloading patool-4.0.0-py2.py3-none-any.whl (86 kB)
                         86.3/86.3 kB
2.3 MB/s eta 0:00:00
Installing collected packages: patool
Successfully installed patool-4.0.0
```

[23]:
```
# Importing the library to read the zip files
import zipfile
with zipfile.ZipFile("/content/drive/MyDrive/cats_vs_dogs_small.zip","r") as␣
 ↪dataset_zip:
    dataset_zip.extractall("/content") #Extracting the required files to␣
 ↪specific directory
```

[25]:
```
# Importing the OS module to manage file paths
import os
dataset_directory = "/content/cats_vs_dogs_small"
print( "Contents of the base directory:", os.listdir(dataset_directory))
```

```
Contents of the base directory: ['test', 'validation', 'train']
```

[28]:
```
pip install tensorflow
```

```
Collecting tensorflow
  Downloading tensorflow-2.19.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_
x86_64.whl.metadata (4.1 kB)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (1.4.0)
```

```
Collecting astunparse>=1.6.0 (from tensorflow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=24.3.25 (from tensorflow)
  Downloading flatbuffers-25.2.10-py2.py3-none-any.whl.metadata (875 bytes)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6.0)
Collecting google-pasta>=0.1.1 (from tensorflow)
  Downloading google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting libclang>=13.0.0 (from tensorflow)
  Downloading libclang-18.1.1-py2.py3-none-manylinux2010_x86_64.whl.metadata
(5.2 kB)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (24.2)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3
in /usr/local/lib/python3.11/dist-packages (from tensorflow) (5.29.4)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)
Collecting tensorboard~=2.19.0 (from tensorflow)
  Downloading tensorboard-2.19.0-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (3.13.0)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (0.5.1)
Collecting tensorflow-io-gcs-filesystem>=0.23.1 (from tensorflow)
  Downloading tensorflow_io_gcs_filesystem-0.37.1-cp311-cp311-manylinux_2_17_x86
_64.manylinux2014_x86_64.whl.metadata (14 kB)
Collecting wheel<1.0,>=0.23.0 (from astunparse>=1.6.0->tensorflow)
  Downloading wheel-0.45.1-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages
```

```
(from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages
(from keras>=3.5.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages
(from keras>=3.5.0->tensorflow) (0.14.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
(2025.1.31)
Requirement already satisfied: markdown>=2.6.8 in /usr/lib/python3/dist-packages
(from tensorboard~=2.19.0->tensorflow) (3.3.6)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from
tensorboard~=2.19.0->tensorflow)
  Downloading tensorboard_data_server-0.7.2-py3-none-
manylinux_2_31_x86_64.whl.metadata (1.1 kB)
Collecting werkzeug>=1.0.1 (from tensorboard~=2.19.0->tensorflow)
  Downloading werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.11/dist-packages (from
werkzeug>=1.0.1->tensorboard~=2.19.0->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow)
(3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow)
(2.19.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-
packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.2)
Downloading
tensorflow-2.19.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(644.9 MB)
                          644.9/644.9 MB
1.4 MB/s eta 0:00:00
Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Downloading flatbuffers-25.2.10-py2.py3-none-any.whl (30 kB)
Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
                          57.5/57.5 kB
3.1 MB/s eta 0:00:00
Downloading libclang-18.1.1-py2.py3-none-manylinux2010_x86_64.whl (24.5
MB)
                          24.5/24.5 MB
```

```
72.1 MB/s eta 0:00:00
Downloading tensorboard-2.19.0-py3-none-any.whl (5.5 MB)
                         5.5/5.5 MB
99.6 MB/s eta 0:00:00
Downloading tensorflow_io_gcs_filesystem-0.37.1-cp311-cp311-manylinux_2_17
_x86_64.manylinux2014_x86_64.whl (5.1 MB)
                         5.1/5.1 MB
101.9 MB/s eta 0:00:00
Downloading tensorboard_data_server-0.7.2-py3-none-
manylinux_2_31_x86_64.whl (6.6 MB)
                         6.6/6.6 MB
106.2 MB/s eta 0:00:00
Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)
                         224.5/224.5 kB
16.6 MB/s eta 0:00:00
Downloading wheel-0.45.1-py3-none-any.whl (72 kB)
                         72.5/72.5 kB
5.6 MB/s eta 0:00:00
Installing collected packages: libclang, flatbuffers, wheel, werkzeug,
tensorflow-io-gcs-filesystem, tensorboard-data-server, google-pasta,
tensorboard, astunparse, tensorflow
Successfully installed astunparse-1.6.3 flatbuffers-25.2.10 google-pasta-0.2.0
libclang-18.1.1 tensorboard-2.19.0 tensorboard-data-server-0.7.2
tensorflow-2.19.0 tensorflow-io-gcs-filesystem-0.37.1 werkzeug-3.1.3
wheel-0.45.1
```

[29]:
```python
import os
import pathlib # To change file system paths
import numpy as np  # Importing numpy library
import matplotlib.pyplot as plotlib # Importing library for visualizations
import tensorflow as tf  # Importing the TensorFlow library
from tensorflow import keras # Importing necessary modules from Keras and␣
 ↪TensorFlow libraries
from tensorflow.keras import layers, models, applications # Importing layers,␣
 ↪models, and pre-trained applications from tensorflow and keras

# Setting the correct path to your dataset for Training, Validation and Testing
path_train = '/content/cats_vs_dogs_small/train'
path_val = '/content/cats_vs_dogs_small/validation'
path_test = '/content/cats_vs_dogs_small/test'
```

[30]:
```python
# Loading dataset images from training to Tesnor flow dataset
data_train = keras.preprocessing.image_dataset_from_directory(
    path_train,
    image_size=(180, 180),
    batch_size=32)
```

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

[8]:
```
# Loading Validation Images
data_val = keras.preprocessing.image_dataset_from_directory(
    path_val,
    image_size=(180, 180),
    batch_size=32)
```

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

[9]:
```
# Loading Testing images
data_test = keras.preprocessing.image_dataset_from_directory(
    path_test,
    image_size=(180, 180),
    batch_size=32)
```

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

[31]:
```
# Displaying top 9 images in a grid format from the dataset
def show_images(dataset):
    plotlib.figure(figsize=(10, 10)) #mentioning the size of image required
    for images, classifier_model_image_labels in dataset.take(1):
        for i in range(9):
            ax = plotlib.subplot(3, 3, i + 1)
            plotlib.imshow(images[i].numpy().astype("uint8")) #Displays images
            plotlib.title("Cat" if classifier_model_image_labels[i] == 0 else␣
 ↪"Dog")
            plotlib.axis("off")
    plotlib.show()
```

[32]:
```
# Calling the function top 9 random display images
show_images(data_train)
```

[13]:
```python
def build_custom_classifier_model():
     #Function to build the CNN model
    cnn_classifier_model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=(180, 180, 3)),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(128, (3, 3), activation='relu'),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.Dense(1, activation='sigmoid')
    ])

    cnn_classifier_model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return cnn_classifier_model
```

[14]:
```
# Data augmentation and preprocessing with sample size parameters
def prepare_data_flows(path_train, path_val, batch_size, num_samples=None):
    train_datagen = keras.preprocessing.image.ImageDataGenerator(
        rescale=1./255,
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest'
    )
    val_datagen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

    gen_train = train_datagen.flow_from_directory(
        path_train,
        target_size=(180, 180),
        batch_size=batch_size,
        class_mode='binary'
    )
    val_generator = val_datagen.flow_from_directory(
        path_val,
        target_size=(180, 180),
        batch_size=batch_size,
        class_mode='binary'
    )
    return gen_train, val_generator
```

[15]:
```
# Training the cnn_classifier_model
def fit_classifier_model(cnn_classifier_model, gen_train, gen_val, epochs=30):␣
 ↪# Change: Rename 'num_epochs' to 'epochs'
    training_log = cnn_classifier_model.fit(gen_train,
                        validation_data=gen_val,
                                epochs=epochs)  # Change: Use 'epochs' instead␣
 ↪of 'num_epochs'
    return training_log
```

[16]:
```
# Step 1: Training the cnn_classifier_model from scratch with 1000 samples
train_generator_1, validation_generator_1 = prepare_data_flows(path_train,␣
 ↪path_val, batch_size=32, num_samples=1000)
```

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

[17]:
```
classifier_model_A = build_custom_classifier_model()
```

```
/usr/local/lib/python3.11/dist-
packages/keras/src/layers/convolutional/base_conv.py:107: 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__(activity_regularizer=activity_regularizer, **kwargs)
```

[18]:
```
training_log_A = fit_classifier_model(classifier_model_A, train_generator_1,␣
 ↪validation_generator_1)
```

```
/usr/local/lib/python3.11/dist-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()

Epoch 1/30
63/63                26s 378ms/step -
accuracy: 0.4900 - loss: 1.2645 - val_accuracy: 0.5470 - val_loss: 0.6891
Epoch 2/30
63/63                24s 373ms/step -
accuracy: 0.5686 - loss: 0.6838 - val_accuracy: 0.6050 - val_loss: 0.6622
Epoch 3/30
63/63                24s 370ms/step -
accuracy: 0.5750 - loss: 0.6725 - val_accuracy: 0.5870 - val_loss: 0.6601
Epoch 4/30
63/63                23s 368ms/step -
accuracy: 0.6191 - loss: 0.6490 - val_accuracy: 0.6270 - val_loss: 0.6637
Epoch 5/30
63/63                24s 371ms/step -
accuracy: 0.5930 - loss: 0.6624 - val_accuracy: 0.5980 - val_loss: 0.6556
Epoch 6/30
63/63                24s 372ms/step -
accuracy: 0.6178 - loss: 0.6577 - val_accuracy: 0.6330 - val_loss: 0.6418
Epoch 7/30
63/63                24s 370ms/step -
accuracy: 0.6131 - loss: 0.6531 - val_accuracy: 0.6110 - val_loss: 0.6422
Epoch 8/30
63/63                24s 375ms/step -
accuracy: 0.6364 - loss: 0.6321 - val_accuracy: 0.6680 - val_loss: 0.5978
Epoch 9/30
```

```
63/63            24s 374ms/step -
accuracy: 0.6509 - loss: 0.6249 - val_accuracy: 0.6520 - val_loss: 0.6096
Epoch 10/30
63/63            23s 369ms/step -
accuracy: 0.6642 - loss: 0.6048 - val_accuracy: 0.6830 - val_loss: 0.5935
Epoch 11/30
63/63            24s 370ms/step -
accuracy: 0.6719 - loss: 0.5955 - val_accuracy: 0.6810 - val_loss: 0.5961
Epoch 12/30
63/63            24s 373ms/step -
accuracy: 0.6697 - loss: 0.5994 - val_accuracy: 0.7010 - val_loss: 0.5500
Epoch 13/30
63/63            23s 367ms/step -
accuracy: 0.7041 - loss: 0.5662 - val_accuracy: 0.7040 - val_loss: 0.5537
Epoch 14/30
63/63            23s 370ms/step -
accuracy: 0.7000 - loss: 0.5633 - val_accuracy: 0.7220 - val_loss: 0.5387
Epoch 15/30
63/63            24s 372ms/step -
accuracy: 0.6909 - loss: 0.5764 - val_accuracy: 0.6590 - val_loss: 0.6505
Epoch 16/30
63/63            23s 367ms/step -
accuracy: 0.6702 - loss: 0.5770 - val_accuracy: 0.7330 - val_loss: 0.5248
Epoch 17/30
63/63            24s 371ms/step -
accuracy: 0.7333 - loss: 0.5388 - val_accuracy: 0.7240 - val_loss: 0.5169
Epoch 18/30
63/63            23s 363ms/step -
accuracy: 0.7188 - loss: 0.5384 - val_accuracy: 0.7310 - val_loss: 0.5476
Epoch 19/30
63/63            23s 369ms/step -
accuracy: 0.7191 - loss: 0.5517 - val_accuracy: 0.7220 - val_loss: 0.5207
Epoch 20/30
63/63            23s 362ms/step -
accuracy: 0.6869 - loss: 0.5722 - val_accuracy: 0.7040 - val_loss: 0.5598
Epoch 21/30
63/63            23s 367ms/step -
accuracy: 0.6989 - loss: 0.5836 - val_accuracy: 0.7360 - val_loss: 0.5334
Epoch 22/30
63/63            24s 371ms/step -
accuracy: 0.7089 - loss: 0.5495 - val_accuracy: 0.7530 - val_loss: 0.4967
Epoch 23/30
63/63            23s 364ms/step -
accuracy: 0.7162 - loss: 0.5572 - val_accuracy: 0.7420 - val_loss: 0.5208
Epoch 24/30
63/63            23s 369ms/step -
accuracy: 0.7649 - loss: 0.5021 - val_accuracy: 0.7270 - val_loss: 0.5436
Epoch 25/30
```

```
63/63              23s 365ms/step -
accuracy: 0.7450 - loss: 0.5117 - val_accuracy: 0.7370 - val_loss: 0.5134
Epoch 26/30
63/63              23s 364ms/step -
accuracy: 0.7724 - loss: 0.4957 - val_accuracy: 0.7500 - val_loss: 0.4890
Epoch 27/30
63/63              24s 373ms/step -
accuracy: 0.7495 - loss: 0.5258 - val_accuracy: 0.7600 - val_loss: 0.4925
Epoch 28/30
63/63              24s 372ms/step -
accuracy: 0.7766 - loss: 0.4763 - val_accuracy: 0.7410 - val_loss: 0.5183
Epoch 29/30
63/63              23s 369ms/step -
accuracy: 0.7610 - loss: 0.4856 - val_accuracy: 0.7500 - val_loss: 0.5007
Epoch 30/30
63/63              24s 371ms/step -
accuracy: 0.7715 - loss: 0.4677 - val_accuracy: 0.7410 - val_loss: 0.5442
```

[19]:
```
# Step 2: Increasing training samples to 1500
train_generator_2, validation_generator_2 = prepare_data_flows(path_train,␣
↪path_val, batch_size=32, num_samples=1500)
classifier_model_B = build_custom_classifier_model()
training_log_B = fit_classifier_model(classifier_model_B, train_generator_2,␣
↪validation_generator_2)
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Epoch 1/30
63/63              25s 367ms/step -
accuracy: 0.5019 - loss: 1.2752 - val_accuracy: 0.5400 - val_loss: 0.6916
Epoch 2/30
63/63              23s 363ms/step -
accuracy: 0.5122 - loss: 0.6929 - val_accuracy: 0.5690 - val_loss: 0.6879
Epoch 3/30
63/63              23s 361ms/step -
accuracy: 0.5480 - loss: 0.6847 - val_accuracy: 0.6430 - val_loss: 0.6420
Epoch 4/30
63/63              23s 368ms/step -
accuracy: 0.6200 - loss: 0.6578 - val_accuracy: 0.6000 - val_loss: 0.6651
Epoch 5/30
63/63              23s 362ms/step -
accuracy: 0.6136 - loss: 0.6574 - val_accuracy: 0.6830 - val_loss: 0.6082
Epoch 6/30
63/63              23s 366ms/step -
accuracy: 0.6270 - loss: 0.6445 - val_accuracy: 0.6280 - val_loss: 0.6232
Epoch 7/30
63/63              23s 364ms/step -
accuracy: 0.6712 - loss: 0.6277 - val_accuracy: 0.7100 - val_loss: 0.5696
```

```
Epoch 8/30
63/63            23s 369ms/step -
accuracy: 0.6949 - loss: 0.5883 - val_accuracy: 0.6800 - val_loss: 0.5775
Epoch 9/30
63/63            23s 362ms/step -
accuracy: 0.6885 - loss: 0.5837 - val_accuracy: 0.7130 - val_loss: 0.5540
Epoch 10/30
63/63            23s 365ms/step -
accuracy: 0.6904 - loss: 0.5780 - val_accuracy: 0.6950 - val_loss: 0.5511
Epoch 11/30
63/63            23s 367ms/step -
accuracy: 0.7118 - loss: 0.5714 - val_accuracy: 0.7210 - val_loss: 0.5531
Epoch 12/30
63/63            23s 364ms/step -
accuracy: 0.7148 - loss: 0.5556 - val_accuracy: 0.7170 - val_loss: 0.5594
Epoch 13/30
63/63            23s 366ms/step -
accuracy: 0.6975 - loss: 0.5700 - val_accuracy: 0.7080 - val_loss: 0.5669
Epoch 14/30
63/63            23s 367ms/step -
accuracy: 0.6983 - loss: 0.5798 - val_accuracy: 0.7550 - val_loss: 0.5160
Epoch 15/30
63/63            23s 362ms/step -
accuracy: 0.6988 - loss: 0.5778 - val_accuracy: 0.7330 - val_loss: 0.5327
Epoch 16/30
63/63            24s 371ms/step -
accuracy: 0.7154 - loss: 0.5447 - val_accuracy: 0.7370 - val_loss: 0.5150
Epoch 17/30
63/63            23s 364ms/step -
accuracy: 0.7455 - loss: 0.5283 - val_accuracy: 0.7120 - val_loss: 0.5458
Epoch 18/30
63/63            23s 362ms/step -
accuracy: 0.7282 - loss: 0.5415 - val_accuracy: 0.7550 - val_loss: 0.5028
Epoch 19/30
63/63            23s 364ms/step -
accuracy: 0.7302 - loss: 0.5383 - val_accuracy: 0.7420 - val_loss: 0.5186
Epoch 20/30
63/63            23s 365ms/step -
accuracy: 0.7217 - loss: 0.5437 - val_accuracy: 0.7450 - val_loss: 0.5073
Epoch 21/30
63/63            23s 363ms/step -
accuracy: 0.7304 - loss: 0.5374 - val_accuracy: 0.7210 - val_loss: 0.5401
Epoch 22/30
63/63            23s 367ms/step -
accuracy: 0.7168 - loss: 0.5518 - val_accuracy: 0.7680 - val_loss: 0.4866
Epoch 23/30
63/63            23s 360ms/step -
accuracy: 0.7267 - loss: 0.5326 - val_accuracy: 0.7310 - val_loss: 0.5252
```

```
Epoch 24/30
63/63                23s 363ms/step -
accuracy: 0.7267 - loss: 0.5464 - val_accuracy: 0.7470 - val_loss: 0.5018
Epoch 25/30
63/63                23s 368ms/step -
accuracy: 0.7579 - loss: 0.5121 - val_accuracy: 0.7550 - val_loss: 0.4989
Epoch 26/30
63/63                23s 366ms/step -
accuracy: 0.7315 - loss: 0.5289 - val_accuracy: 0.7570 - val_loss: 0.4975
Epoch 27/30
63/63                23s 361ms/step -
accuracy: 0.7407 - loss: 0.5111 - val_accuracy: 0.7190 - val_loss: 0.5683
Epoch 28/30
63/63                23s 366ms/step -
accuracy: 0.7254 - loss: 0.5377 - val_accuracy: 0.7500 - val_loss: 0.5263
Epoch 29/30
63/63                23s 364ms/step -
accuracy: 0.7433 - loss: 0.5152 - val_accuracy: 0.7780 - val_loss: 0.4778
Epoch 30/30
63/63                23s 370ms/step -
accuracy: 0.7301 - loss: 0.5286 - val_accuracy: 0.7840 - val_loss: 0.4806
```

[20]:
```
# Step 3: Using the full 2000 samples
train_generator_3, validation_generator_3 = prepare_data_flows(path_train,␣
 ↪path_val, batch_size=32, num_samples=2000)
classifier_model_C = build_custom_classifier_model()
training_log_C = fit_classifier_model(classifier_model_C, train_generator_3,␣
 ↪validation_generator_3)
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Epoch 1/30
63/63                25s 374ms/step -
accuracy: 0.5197 - loss: 0.9573 - val_accuracy: 0.5840 - val_loss: 0.6850
Epoch 2/30
63/63                24s 371ms/step -
accuracy: 0.5417 - loss: 0.6900 - val_accuracy: 0.5040 - val_loss: 0.6944
Epoch 3/30
63/63                23s 364ms/step -
accuracy: 0.5416 - loss: 0.6863 - val_accuracy: 0.5990 - val_loss: 0.6863
Epoch 4/30
63/63                24s 373ms/step -
accuracy: 0.5328 - loss: 0.6866 - val_accuracy: 0.5970 - val_loss: 0.6693
Epoch 5/30
63/63                23s 365ms/step -
accuracy: 0.5803 - loss: 0.6844 - val_accuracy: 0.5950 - val_loss: 0.6708
Epoch 6/30
63/63                24s 376ms/step -
```

```
accuracy: 0.5766 - loss: 0.6759 - val_accuracy: 0.6550 - val_loss: 0.6343
Epoch 7/30
63/63            23s 370ms/step -
accuracy: 0.6412 - loss: 0.6480 - val_accuracy: 0.6460 - val_loss: 0.6224
Epoch 8/30
63/63            23s 367ms/step -
accuracy: 0.6332 - loss: 0.6351 - val_accuracy: 0.6710 - val_loss: 0.6211
Epoch 9/30
63/63            24s 372ms/step -
accuracy: 0.6425 - loss: 0.6422 - val_accuracy: 0.6610 - val_loss: 0.6173
Epoch 10/30
63/63            23s 364ms/step -
accuracy: 0.6067 - loss: 0.6650 - val_accuracy: 0.6540 - val_loss: 0.6115
Epoch 11/30
63/63            23s 370ms/step -
accuracy: 0.6473 - loss: 0.6253 - val_accuracy: 0.6880 - val_loss: 0.5912
Epoch 12/30
63/63            24s 371ms/step -
accuracy: 0.6891 - loss: 0.6062 - val_accuracy: 0.7040 - val_loss: 0.5676
Epoch 13/30
63/63            24s 374ms/step -
accuracy: 0.6840 - loss: 0.5716 - val_accuracy: 0.6840 - val_loss: 0.6545
Epoch 14/30
63/63            23s 368ms/step -
accuracy: 0.7062 - loss: 0.5814 - val_accuracy: 0.6930 - val_loss: 0.5735
Epoch 15/30
63/63            24s 371ms/step -
accuracy: 0.6947 - loss: 0.5772 - val_accuracy: 0.6950 - val_loss: 0.5628
Epoch 16/30
63/63            23s 365ms/step -
accuracy: 0.7111 - loss: 0.5567 - val_accuracy: 0.7140 - val_loss: 0.5664
Epoch 17/30
63/63            24s 371ms/step -
accuracy: 0.6813 - loss: 0.5773 - val_accuracy: 0.7250 - val_loss: 0.5158
Epoch 18/30
63/63            24s 370ms/step -
accuracy: 0.7097 - loss: 0.5562 - val_accuracy: 0.7290 - val_loss: 0.5300
Epoch 19/30
63/63            23s 366ms/step -
accuracy: 0.7409 - loss: 0.5327 - val_accuracy: 0.7130 - val_loss: 0.5525
Epoch 20/30
63/63            23s 363ms/step -
accuracy: 0.7222 - loss: 0.5471 - val_accuracy: 0.7220 - val_loss: 0.5405
Epoch 21/30
63/63            24s 372ms/step -
accuracy: 0.7378 - loss: 0.5418 - val_accuracy: 0.7270 - val_loss: 0.5237
Epoch 22/30
63/63            24s 372ms/step -
```

```
accuracy: 0.7344 - loss: 0.5238 - val_accuracy: 0.7350 - val_loss: 0.5153
Epoch 23/30
63/63                 23s 365ms/step -
accuracy: 0.7319 - loss: 0.5159 - val_accuracy: 0.7450 - val_loss: 0.5022
Epoch 24/30
63/63                 24s 372ms/step -
accuracy: 0.7519 - loss: 0.5109 - val_accuracy: 0.7490 - val_loss: 0.5091
Epoch 25/30
63/63                 23s 361ms/step -
accuracy: 0.7302 - loss: 0.5202 - val_accuracy: 0.7680 - val_loss: 0.4867
Epoch 26/30
63/63                 23s 366ms/step -
accuracy: 0.7573 - loss: 0.4787 - val_accuracy: 0.7590 - val_loss: 0.4851
Epoch 27/30
63/63                 23s 364ms/step -
accuracy: 0.7658 - loss: 0.4961 - val_accuracy: 0.7710 - val_loss: 0.4703
Epoch 28/30
63/63                 23s 358ms/step -
accuracy: 0.7496 - loss: 0.5104 - val_accuracy: 0.7490 - val_loss: 0.4986
Epoch 29/30
63/63                 24s 371ms/step -
accuracy: 0.7680 - loss: 0.4944 - val_accuracy: 0.7650 - val_loss: 0.4767
Epoch 30/30
63/63                 23s 365ms/step -
accuracy: 0.7596 - loss: 0.5048 - val_accuracy: 0.7770 - val_loss: 0.4496
```

[21]:
```
# Step 4: Using a pretrained cnn_classifier_model (e.g., VGG16)
def build_pretrained_classifier_model():
    base_classifier_model = applications.VGG16(include_top=False,␣
 ↪weights='imagenet', input_shape=(180, 180, 3))
    base_classifier_model.trainable = False  # Freeze the base␣
 ↪cnn_classifier_model
```

[29]:
```
def build_pretrained_classifier_model():
    base_classifier_model = applications.VGG16(include_top=False,␣
 ↪weights='imagenet', input_shape=(180, 180, 3))
    base_classifier_model.trainable = False  # Freeze the base␣
 ↪cnn_classifier_model
    cnn_classifier_model = models.Sequential([
        base_classifier_model,
        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.Dense(1, activation='sigmoid')
    ])

    cnn_classifier_model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
```

```
    return cnn_classifier_model
```

[30]:
```
# Repeaing Steps 1-3 with the pretrained cnn_classifier_model
classifier_model_P1 = build_pretrained_classifier_model()
training_log_P1 = fit_classifier_model(classifier_model_P1, train_generator_1,␣
 ↪validation_generator_1)

classifier_model_P2 = build_pretrained_classifier_model()
training_log_P2 = fit_classifier_model(classifier_model_P2, train_generator_2,␣
 ↪validation_generator_2)

classifier_model_P3 = build_pretrained_classifier_model()
training_log_P3 = fit_classifier_model(classifier_model_P3, train_generator_3,␣
 ↪validation_generator_3)
```

```
Epoch 1/30
63/63                 43s 657ms/step -
accuracy: 0.6027 - loss: 1.4969 - val_accuracy: 0.8750 - val_loss: 0.2916
Epoch 2/30
63/63                 40s 639ms/step -
accuracy: 0.8428 - loss: 0.3761 - val_accuracy: 0.8990 - val_loss: 0.2494
Epoch 3/30
63/63                 41s 654ms/step -
accuracy: 0.8576 - loss: 0.3137 - val_accuracy: 0.9030 - val_loss: 0.2307
Epoch 4/30
63/63                 41s 652ms/step -
accuracy: 0.8557 - loss: 0.3281 - val_accuracy: 0.8540 - val_loss: 0.3204
Epoch 5/30
63/63                 41s 654ms/step -
accuracy: 0.8484 - loss: 0.3287 - val_accuracy: 0.9100 - val_loss: 0.2138
Epoch 6/30
63/63                 40s 640ms/step -
accuracy: 0.8698 - loss: 0.2997 - val_accuracy: 0.9140 - val_loss: 0.2172
Epoch 7/30
63/63                 41s 647ms/step -
accuracy: 0.8834 - loss: 0.2648 - val_accuracy: 0.9040 - val_loss: 0.2282
Epoch 8/30
63/63                 41s 651ms/step -
accuracy: 0.8867 - loss: 0.2551 - val_accuracy: 0.9280 - val_loss: 0.2040
Epoch 9/30
63/63                 42s 658ms/step -
accuracy: 0.8932 - loss: 0.2492 - val_accuracy: 0.9200 - val_loss: 0.2093
Epoch 10/30
63/63                 41s 657ms/step -
accuracy: 0.9035 - loss: 0.2388 - val_accuracy: 0.9170 - val_loss: 0.2054
Epoch 11/30
63/63                 41s 653ms/step -
```

```
accuracy: 0.8690 - loss: 0.2854 - val_accuracy: 0.9180 - val_loss: 0.2116
Epoch 12/30
63/63            40s 637ms/step -
accuracy: 0.8887 - loss: 0.2526 - val_accuracy: 0.9130 - val_loss: 0.2253
Epoch 13/30
63/63            40s 638ms/step -
accuracy: 0.8702 - loss: 0.2865 - val_accuracy: 0.9150 - val_loss: 0.2295
Epoch 14/30
63/63            41s 654ms/step -
accuracy: 0.8883 - loss: 0.2568 - val_accuracy: 0.9170 - val_loss: 0.2103
Epoch 15/30
63/63            41s 643ms/step -
accuracy: 0.9007 - loss: 0.2297 - val_accuracy: 0.9110 - val_loss: 0.2138
Epoch 16/30
63/63            40s 630ms/step -
accuracy: 0.8977 - loss: 0.2382 - val_accuracy: 0.9140 - val_loss: 0.2076
Epoch 17/30
63/63            41s 648ms/step -
accuracy: 0.8952 - loss: 0.2336 - val_accuracy: 0.9110 - val_loss: 0.2223
Epoch 18/30
63/63            40s 638ms/step -
accuracy: 0.9112 - loss: 0.2384 - val_accuracy: 0.9190 - val_loss: 0.2070
Epoch 19/30
63/63            40s 629ms/step -
accuracy: 0.8904 - loss: 0.2484 - val_accuracy: 0.9190 - val_loss: 0.2114
Epoch 20/30
63/63            40s 640ms/step -
accuracy: 0.8999 - loss: 0.2308 - val_accuracy: 0.9200 - val_loss: 0.2108
Epoch 21/30
63/63            41s 655ms/step -
accuracy: 0.9014 - loss: 0.2265 - val_accuracy: 0.9000 - val_loss: 0.2223
Epoch 22/30
63/63            40s 638ms/step -
accuracy: 0.8977 - loss: 0.2432 - val_accuracy: 0.9040 - val_loss: 0.2375
Epoch 23/30
63/63            41s 654ms/step -
accuracy: 0.9021 - loss: 0.2405 - val_accuracy: 0.9100 - val_loss: 0.2196
Epoch 24/30
63/63            40s 638ms/step -
accuracy: 0.9095 - loss: 0.2216 - val_accuracy: 0.9080 - val_loss: 0.2340
Epoch 25/30
63/63            41s 651ms/step -
accuracy: 0.9036 - loss: 0.2316 - val_accuracy: 0.9070 - val_loss: 0.2380
Epoch 26/30
63/63            40s 635ms/step -
accuracy: 0.9120 - loss: 0.2133 - val_accuracy: 0.9110 - val_loss: 0.2391
Epoch 27/30
63/63            40s 633ms/step -
```

```
accuracy: 0.9098 - loss: 0.2115 - val_accuracy: 0.9160 - val_loss: 0.2099
Epoch 28/30
63/63            41s 653ms/step -
accuracy: 0.9038 - loss: 0.2171 - val_accuracy: 0.9130 - val_loss: 0.2359
Epoch 29/30
63/63            39s 626ms/step -
accuracy: 0.9007 - loss: 0.2427 - val_accuracy: 0.9120 - val_loss: 0.2284
Epoch 30/30
63/63            41s 658ms/step -
accuracy: 0.9002 - loss: 0.2189 - val_accuracy: 0.9160 - val_loss: 0.2219
Epoch 1/30
63/63            43s 656ms/step -
accuracy: 0.6165 - loss: 1.4961 - val_accuracy: 0.8810 - val_loss: 0.2862
Epoch 2/30
63/63            41s 643ms/step -
accuracy: 0.8262 - loss: 0.3799 - val_accuracy: 0.8890 - val_loss: 0.2406
Epoch 3/30
63/63            41s 647ms/step -
accuracy: 0.8329 - loss: 0.3578 - val_accuracy: 0.9020 - val_loss: 0.2314
Epoch 4/30
63/63            42s 659ms/step -
accuracy: 0.8364 - loss: 0.3329 - val_accuracy: 0.9090 - val_loss: 0.2240
Epoch 5/30
63/63            41s 648ms/step -
accuracy: 0.8420 - loss: 0.3259 - val_accuracy: 0.9100 - val_loss: 0.2132
Epoch 6/30
63/63            41s 651ms/step -
accuracy: 0.8779 - loss: 0.2927 - val_accuracy: 0.9010 - val_loss: 0.2533
Epoch 7/30
63/63            42s 666ms/step -
accuracy: 0.8752 - loss: 0.2806 - val_accuracy: 0.9110 - val_loss: 0.2125
Epoch 8/30
63/63            42s 665ms/step -
accuracy: 0.8902 - loss: 0.2754 - val_accuracy: 0.9000 - val_loss: 0.2222
Epoch 9/30
63/63            41s 642ms/step -
accuracy: 0.8947 - loss: 0.2481 - val_accuracy: 0.9090 - val_loss: 0.2116
Epoch 10/30
63/63            40s 640ms/step -
accuracy: 0.8944 - loss: 0.2592 - val_accuracy: 0.9130 - val_loss: 0.2132
Epoch 11/30
63/63            40s 641ms/step -
accuracy: 0.8911 - loss: 0.2630 - val_accuracy: 0.9040 - val_loss: 0.2277
Epoch 12/30
63/63            40s 639ms/step -
accuracy: 0.8867 - loss: 0.2698 - val_accuracy: 0.9120 - val_loss: 0.2089
Epoch 13/30
63/63            41s 645ms/step -
```

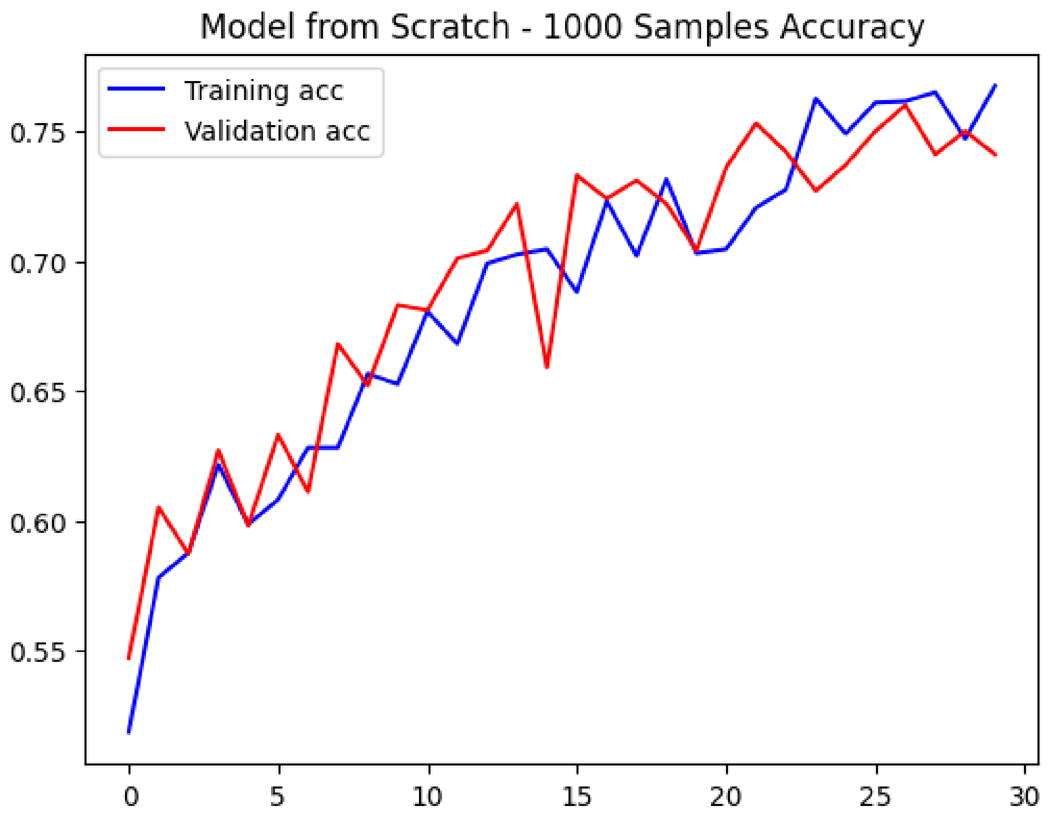
```
accuracy: 0.8943 - loss: 0.2430 - val_accuracy: 0.9070 - val_loss: 0.2373
Epoch 14/30
63/63            41s 644ms/step -
accuracy: 0.8909 - loss: 0.2490 - val_accuracy: 0.9030 - val_loss: 0.2318
Epoch 15/30
63/63            41s 657ms/step -
accuracy: 0.9085 - loss: 0.2372 - val_accuracy: 0.8960 - val_loss: 0.2367
Epoch 16/30
63/63            41s 652ms/step -
accuracy: 0.8991 - loss: 0.2511 - val_accuracy: 0.9020 - val_loss: 0.2193
Epoch 17/30
63/63            41s 649ms/step -
accuracy: 0.8916 - loss: 0.2436 - val_accuracy: 0.9080 - val_loss: 0.2192
Epoch 18/30
63/63            41s 656ms/step -
accuracy: 0.9088 - loss: 0.2213 - val_accuracy: 0.9040 - val_loss: 0.2225
Epoch 19/30
63/63            41s 654ms/step -
accuracy: 0.9061 - loss: 0.2233 - val_accuracy: 0.9070 - val_loss: 0.2261
Epoch 20/30
63/63            41s 650ms/step -
accuracy: 0.9077 - loss: 0.2108 - val_accuracy: 0.8890 - val_loss: 0.2769
Epoch 21/30
63/63            40s 637ms/step -
accuracy: 0.8760 - loss: 0.2416 - val_accuracy: 0.9090 - val_loss: 0.2342
Epoch 22/30
63/63            41s 642ms/step -
accuracy: 0.8839 - loss: 0.2575 - val_accuracy: 0.8950 - val_loss: 0.2678
Epoch 23/30
63/63            41s 644ms/step -
accuracy: 0.9028 - loss: 0.2209 - val_accuracy: 0.9020 - val_loss: 0.2212
Epoch 24/30
63/63            41s 652ms/step -
accuracy: 0.9117 - loss: 0.2224 - val_accuracy: 0.9160 - val_loss: 0.2117
Epoch 25/30
63/63            41s 646ms/step -
accuracy: 0.9067 - loss: 0.2073 - val_accuracy: 0.8940 - val_loss: 0.2730
Epoch 26/30
63/63            41s 656ms/step -
accuracy: 0.8886 - loss: 0.2570 - val_accuracy: 0.8920 - val_loss: 0.2496
Epoch 27/30
63/63            41s 649ms/step -
accuracy: 0.9046 - loss: 0.2139 - val_accuracy: 0.9130 - val_loss: 0.2169
Epoch 28/30
63/63            40s 641ms/step -
accuracy: 0.8946 - loss: 0.2129 - val_accuracy: 0.9090 - val_loss: 0.2129
Epoch 29/30
63/63            41s 644ms/step -
```

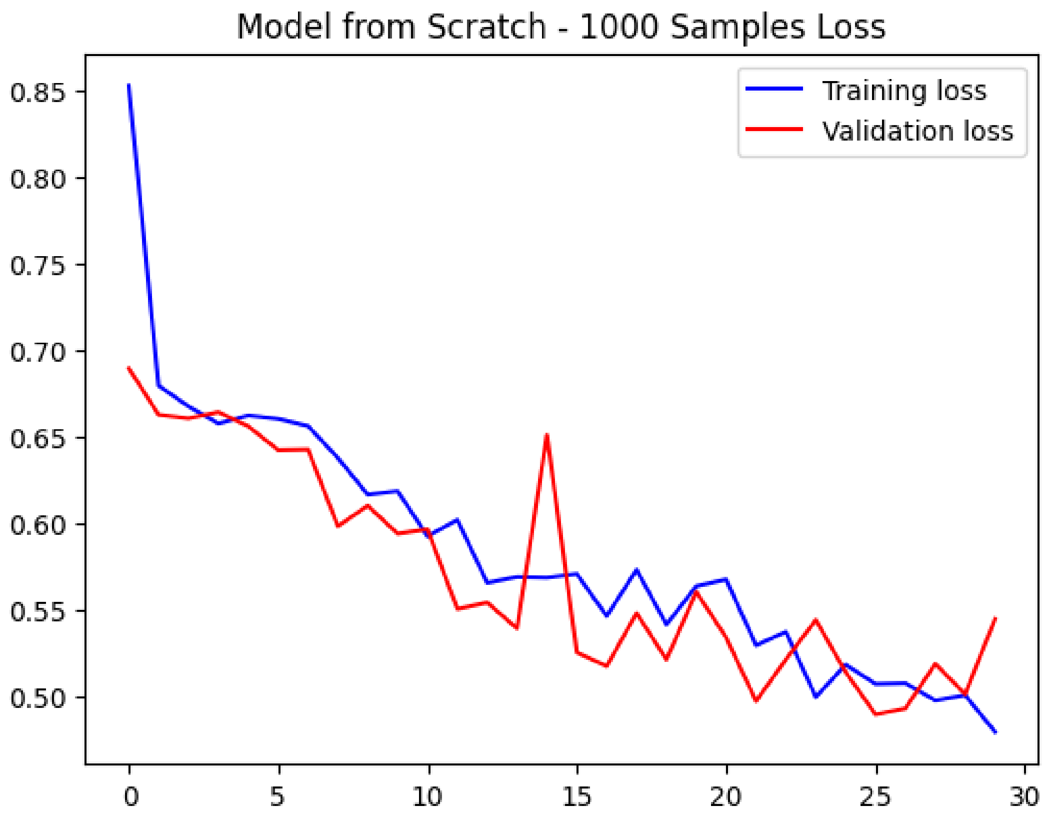
```
accuracy: 0.9073 - loss: 0.2200 - val_accuracy: 0.9060 - val_loss: 0.2190
Epoch 30/30
63/63            41s 647ms/step -
accuracy: 0.8880 - loss: 0.2426 - val_accuracy: 0.9050 - val_loss: 0.2270
Epoch 1/30
63/63            42s 652ms/step -
accuracy: 0.6767 - loss: 1.1598 - val_accuracy: 0.8290 - val_loss: 0.3919
Epoch 2/30
63/63            41s 655ms/step -
accuracy: 0.8295 - loss: 0.3833 - val_accuracy: 0.8990 - val_loss: 0.2495
Epoch 3/30
63/63            41s 649ms/step -
accuracy: 0.8445 - loss: 0.3341 - val_accuracy: 0.8920 - val_loss: 0.2687
Epoch 4/30
63/63            41s 649ms/step -
accuracy: 0.8760 - loss: 0.3086 - val_accuracy: 0.8870 - val_loss: 0.2550
Epoch 5/30
63/63            40s 640ms/step -
accuracy: 0.8619 - loss: 0.3205 - val_accuracy: 0.9000 - val_loss: 0.2374
Epoch 6/30
63/63            41s 656ms/step -
accuracy: 0.8617 - loss: 0.3046 - val_accuracy: 0.9070 - val_loss: 0.2384
Epoch 7/30
63/63            42s 661ms/step -
accuracy: 0.8548 - loss: 0.3243 - val_accuracy: 0.9050 - val_loss: 0.2237
Epoch 8/30
63/63            41s 646ms/step -
accuracy: 0.8648 - loss: 0.2988 - val_accuracy: 0.9180 - val_loss: 0.2176
Epoch 9/30
63/63            42s 662ms/step -
accuracy: 0.8730 - loss: 0.2767 - val_accuracy: 0.8950 - val_loss: 0.2723
Epoch 10/30
63/63            41s 645ms/step -
accuracy: 0.8796 - loss: 0.2802 - val_accuracy: 0.8740 - val_loss: 0.2994
Epoch 11/30
63/63            41s 656ms/step -
accuracy: 0.8847 - loss: 0.2832 - val_accuracy: 0.8950 - val_loss: 0.2585
Epoch 12/30
63/63            40s 640ms/step -
accuracy: 0.8811 - loss: 0.2486 - val_accuracy: 0.9060 - val_loss: 0.2289
Epoch 13/30
63/63            41s 643ms/step -
accuracy: 0.8887 - loss: 0.2562 - val_accuracy: 0.8960 - val_loss: 0.2266
Epoch 14/30
63/63            41s 651ms/step -
accuracy: 0.8990 - loss: 0.2315 - val_accuracy: 0.9040 - val_loss: 0.2243
Epoch 15/30
63/63            41s 643ms/step -
```

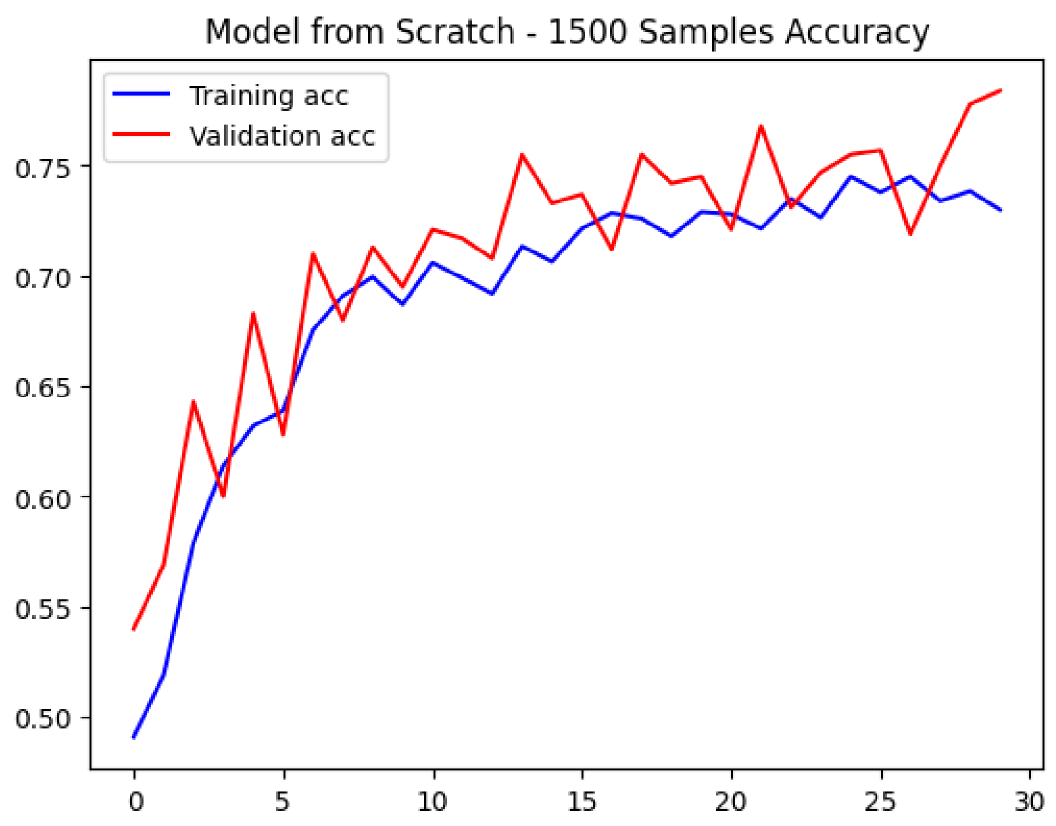
```
accuracy: 0.8983 - loss: 0.2454 - val_accuracy: 0.9030 - val_loss: 0.2397
Epoch 16/30
63/63            41s 649ms/step -
accuracy: 0.9035 - loss: 0.2216 - val_accuracy: 0.9000 - val_loss: 0.2372
Epoch 17/30
63/63            41s 644ms/step -
accuracy: 0.9038 - loss: 0.2469 - val_accuracy: 0.9100 - val_loss: 0.2386
Epoch 18/30
63/63            40s 641ms/step -
accuracy: 0.9059 - loss: 0.2355 - val_accuracy: 0.8730 - val_loss: 0.3208
Epoch 19/30
63/63            40s 632ms/step -
accuracy: 0.8829 - loss: 0.2671 - val_accuracy: 0.9070 - val_loss: 0.2136
Epoch 20/30
63/63            40s 641ms/step -
accuracy: 0.9143 - loss: 0.2122 - val_accuracy: 0.9160 - val_loss: 0.2278
Epoch 21/30
63/63            41s 657ms/step -
accuracy: 0.9065 - loss: 0.2211 - val_accuracy: 0.9000 - val_loss: 0.2299
Epoch 22/30
63/63            40s 641ms/step -
accuracy: 0.9005 - loss: 0.2420 - val_accuracy: 0.9110 - val_loss: 0.2192
Epoch 23/30
63/63            40s 637ms/step -
accuracy: 0.9014 - loss: 0.2197 - val_accuracy: 0.9070 - val_loss: 0.2220
Epoch 24/30
63/63            41s 643ms/step -
accuracy: 0.9196 - loss: 0.1926 - val_accuracy: 0.8910 - val_loss: 0.2481
Epoch 25/30
63/63            41s 645ms/step -
accuracy: 0.8936 - loss: 0.2523 - val_accuracy: 0.9100 - val_loss: 0.2245
Epoch 26/30
63/63            41s 651ms/step -
accuracy: 0.8988 - loss: 0.2301 - val_accuracy: 0.9050 - val_loss: 0.2286
Epoch 27/30
63/63            40s 641ms/step -
accuracy: 0.9117 - loss: 0.2155 - val_accuracy: 0.9080 - val_loss: 0.2204
Epoch 28/30
63/63            40s 636ms/step -
accuracy: 0.9145 - loss: 0.1933 - val_accuracy: 0.9050 - val_loss: 0.2407
Epoch 29/30
63/63            41s 643ms/step -
accuracy: 0.8900 - loss: 0.2542 - val_accuracy: 0.9050 - val_loss: 0.2362
Epoch 30/30
63/63            41s 652ms/step -
accuracy: 0.8990 - loss: 0.2304 - val_accuracy: 0.9090 - val_loss: 0.2215
```

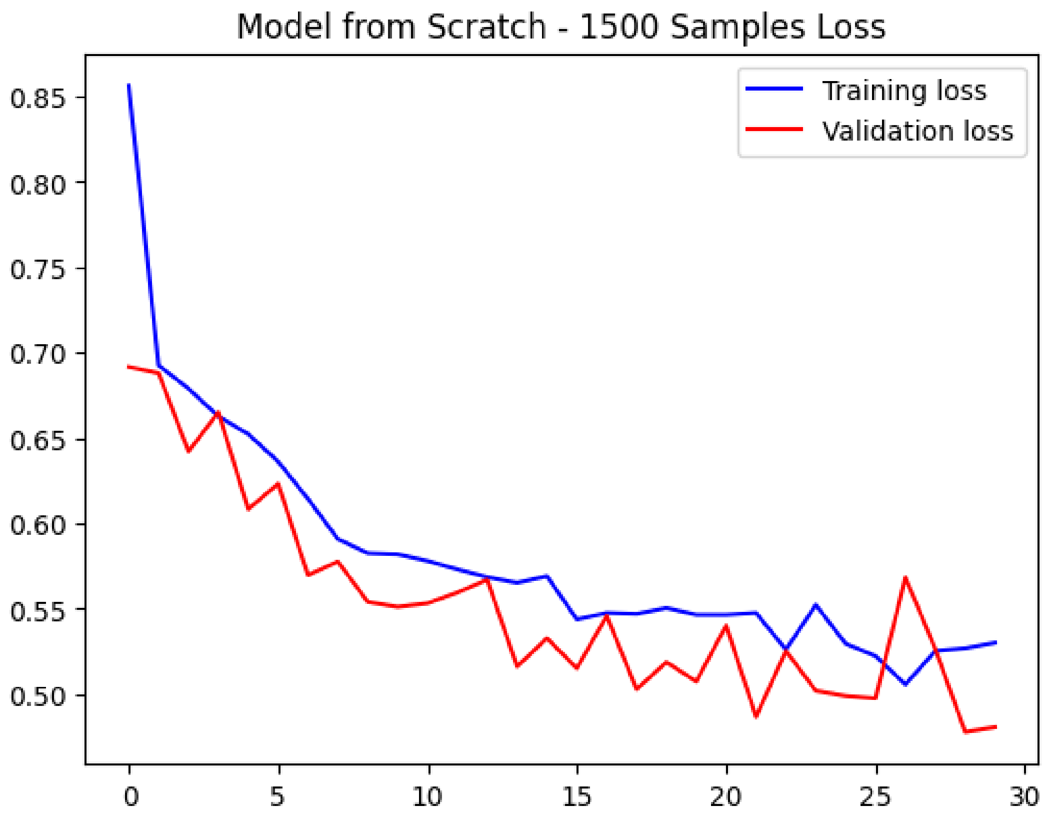
[33]:
```python
# Performance visualization function
def visualize_performance(training_log, title):
    acc = training_log.history['accuracy']
    val_acc = training_log.history['val_accuracy']
    train_loss = training_log.history['loss']
    validation_train_loss = training_log.history['val_loss']

    num_epochs = range(len(acc))

    plotlib.figure()
    plotlib.plot(num_epochs, acc, 'b', label='Training acc')
    plotlib.plot(num_epochs, val_acc, 'r', label='Validation acc')
    plotlib.title(title + ' Accuracy')
    plotlib.legend()

    plotlib.figure()
    plotlib.plot(num_epochs, train_loss, 'b', label='Training loss')
    plotlib.plot(num_epochs, validation_train_loss, 'r', label='Validation␣
 ↪loss')
    plotlib.title(title + ' Loss')
    plotlib.legend()
    plotlib.show()
```

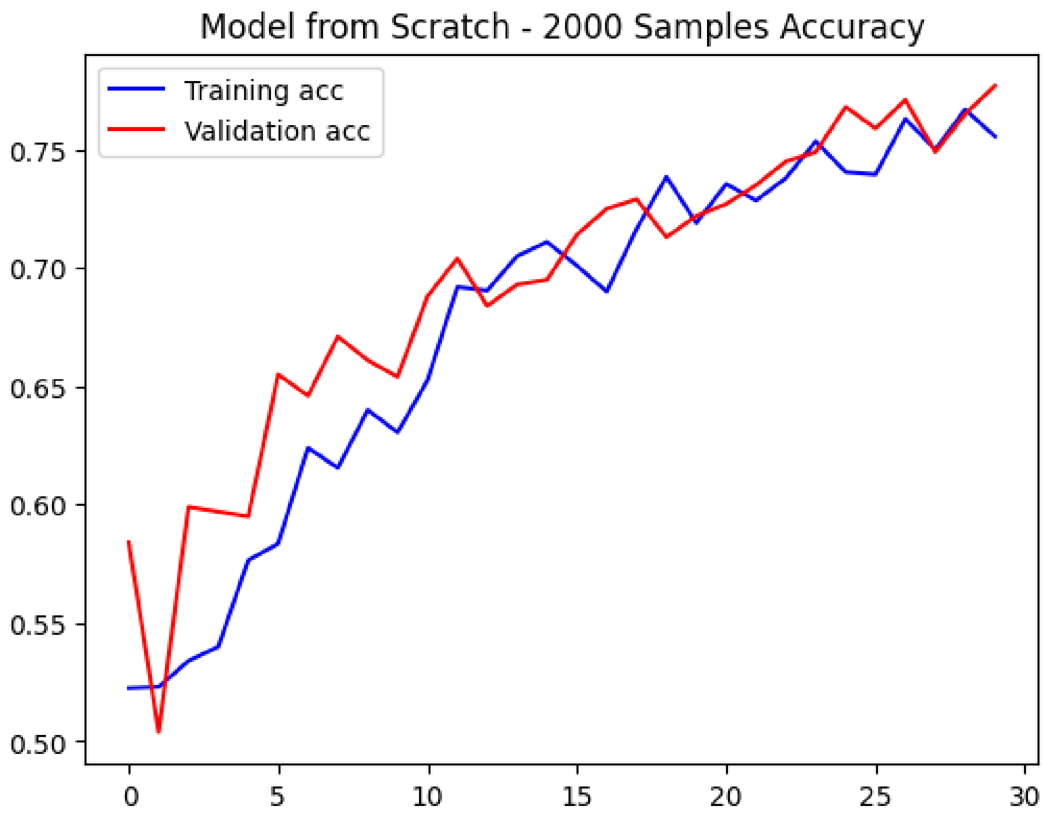
[34]:
```python
# Plotting performance for each cnn_classifier_model
visualize_performance(training_log_A, 'Model from Scratch - 1000 Samples')
visualize_performance(training_log_B, 'Model from Scratch - 1500 Samples')
visualize_performance(training_log_C, 'Model from Scratch - 2000 Samples')
visualize_performance(training_log_P1, 'Pretrained Model - 1000 Samples')
visualize_performance(training_log_P2, 'Pretrained Model - 1500 Samples')
visualize_performance(training_log_P3, 'Pretrained Model - 2000 Samples')
```

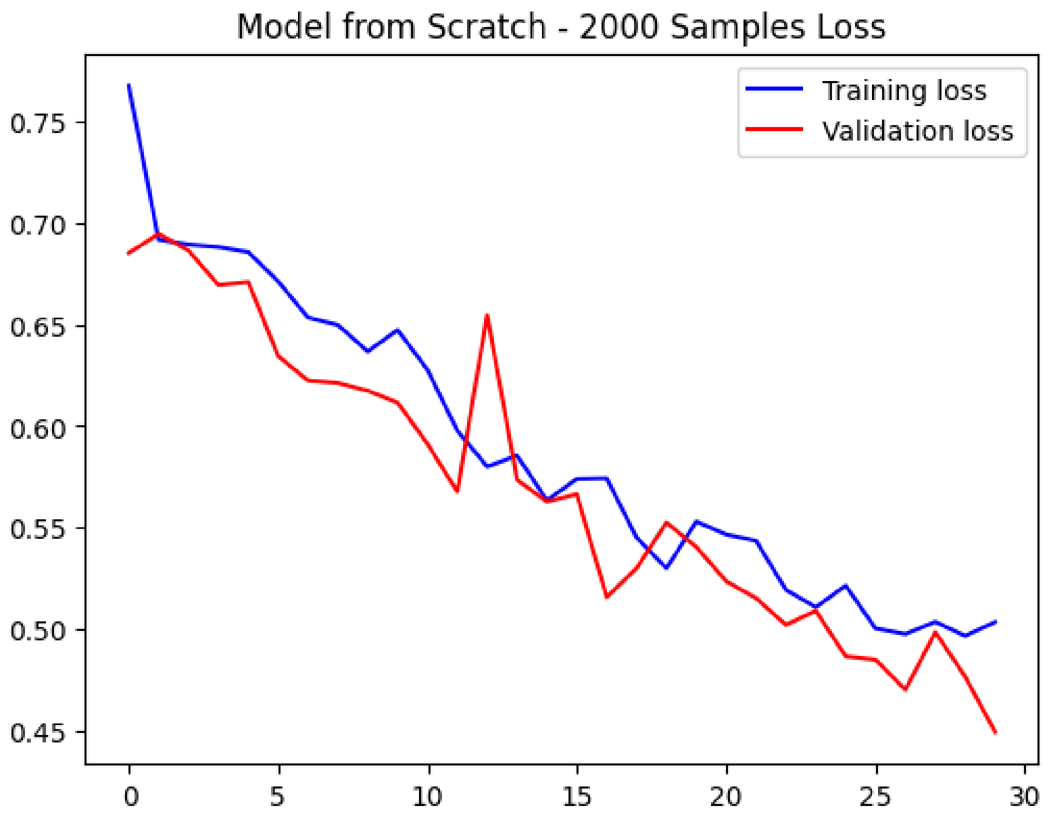

Model from Scratch - 1000 Samples Accuracy
Training acc
Validation acc
0.75
0.70
0.65
0.60
0.55
0
5
10
15
20
25
30

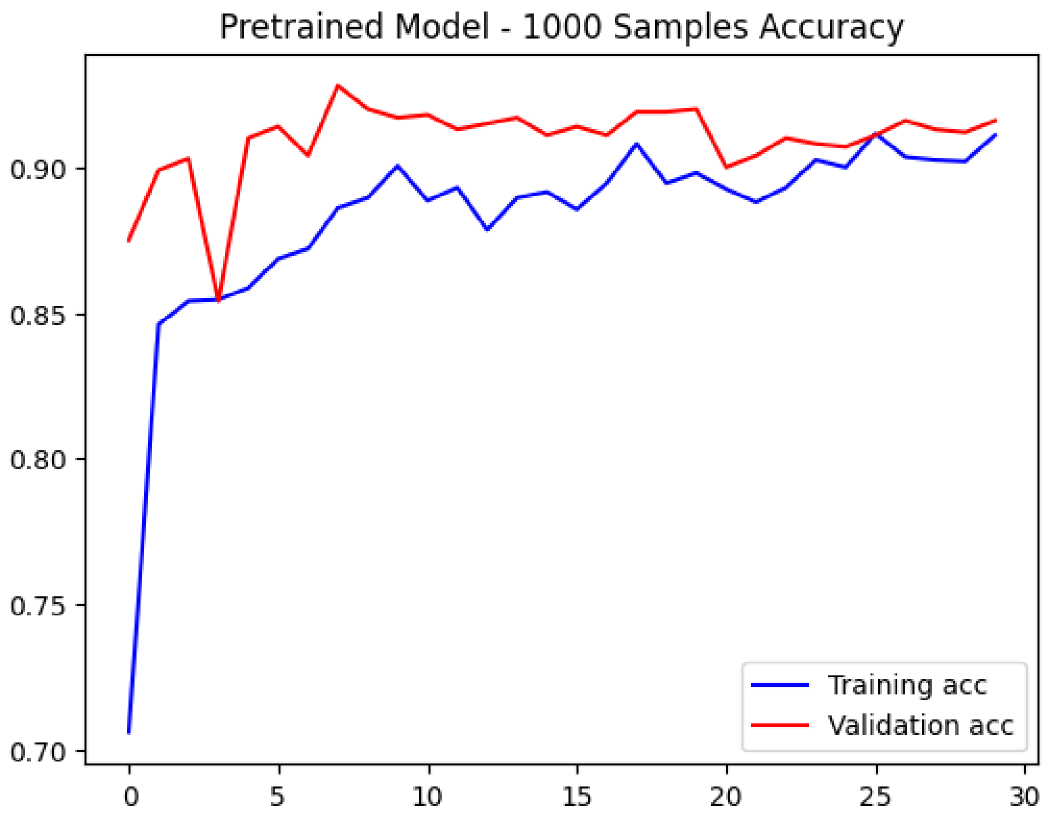

Model from Scratch - 1000 Samples Loss
Training loss
Validation loss
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30

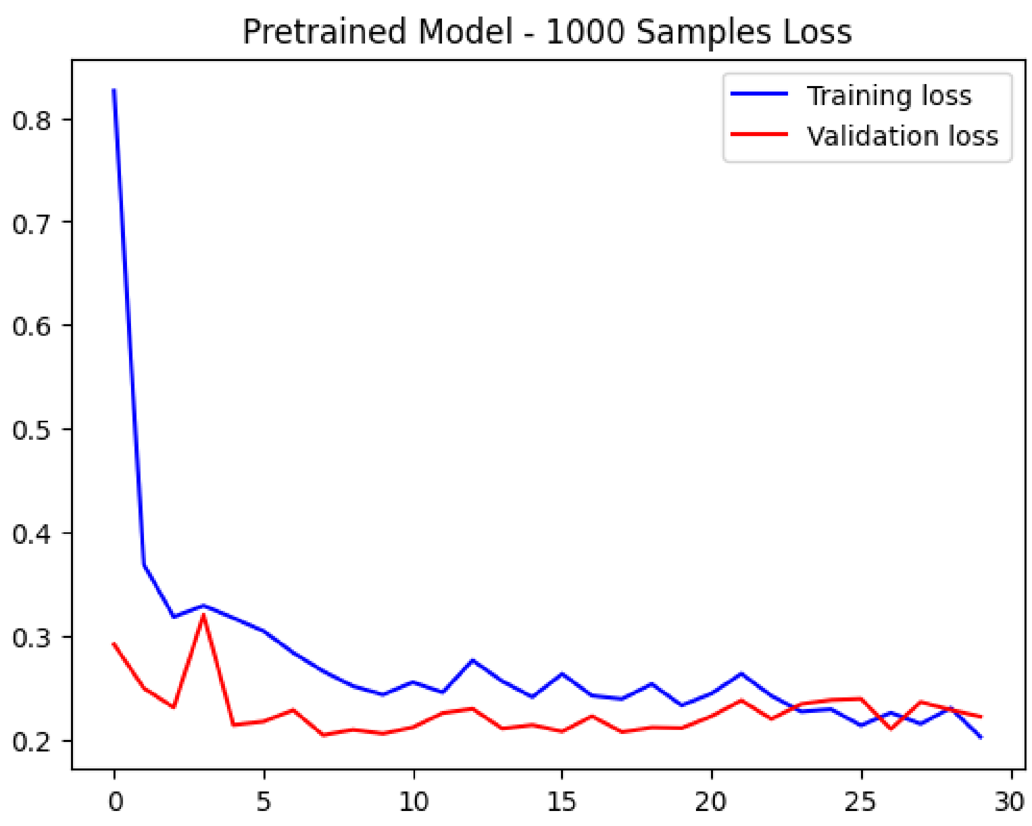

Model from Scratch - 1500 Samples Accuracy
Training acc
Validation acc
0.75
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30

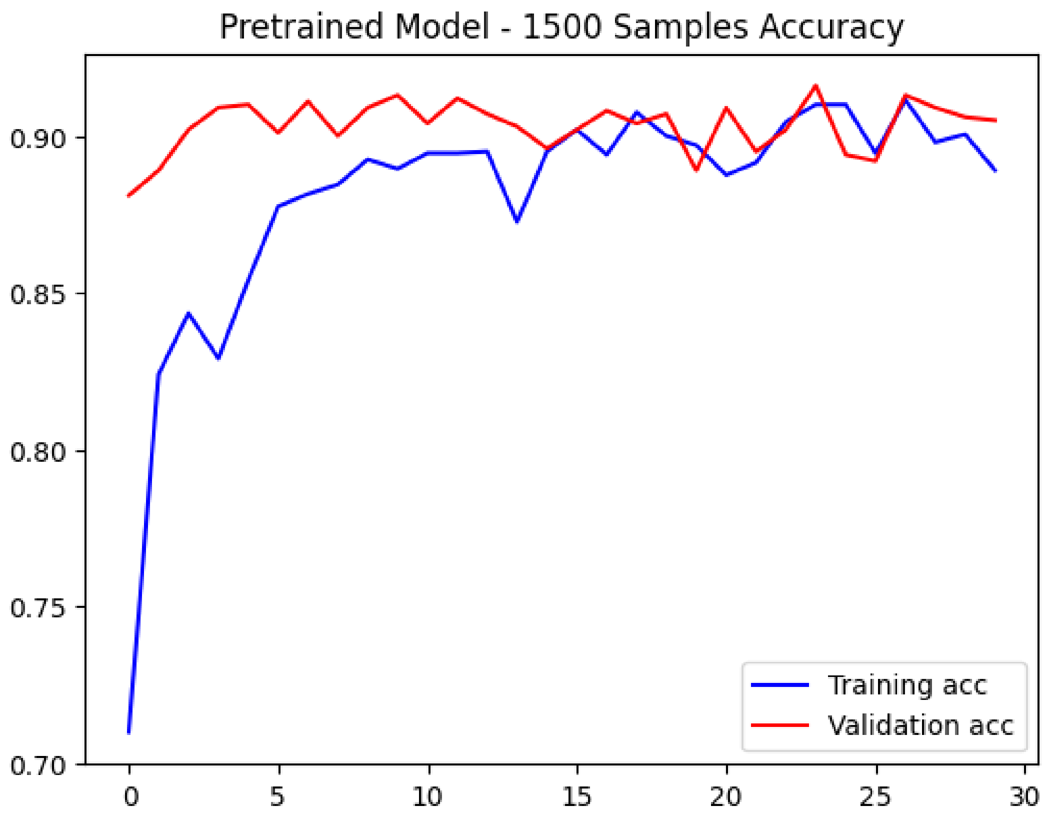

Model from Scratch - 1500 Samples Loss
Training loss
Validation loss
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30

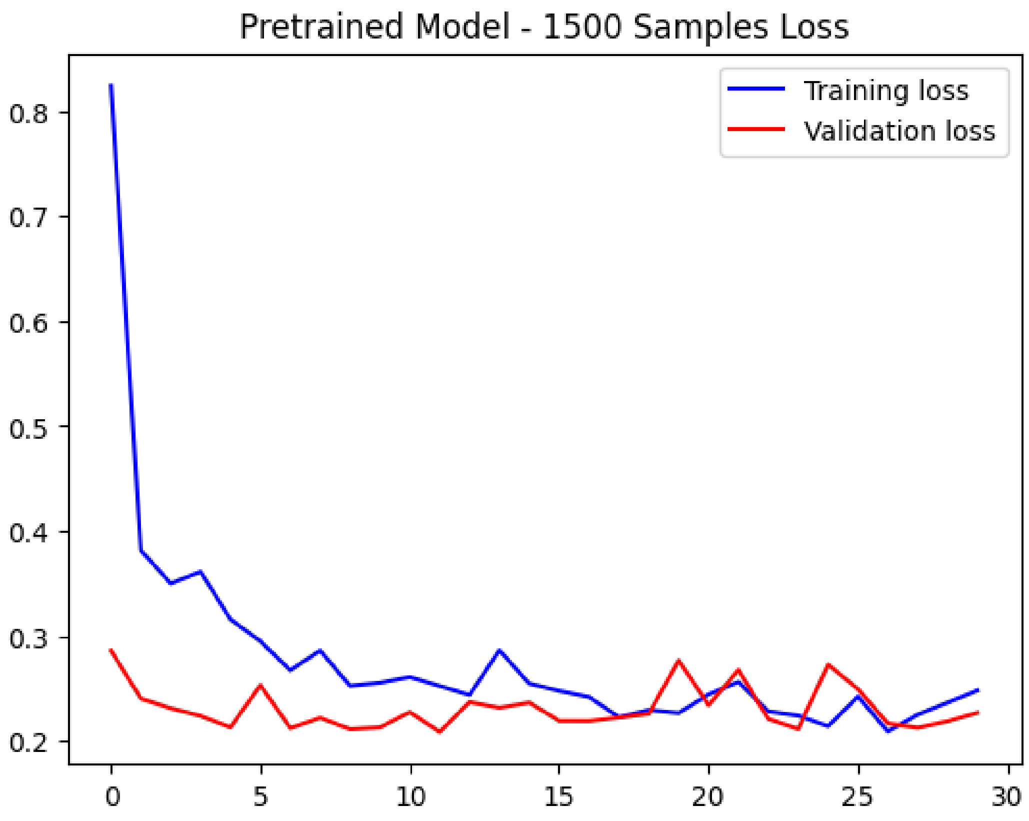

Model from Scratch - 2000 Samples Accuracy
Training acc
Validation acc
0.75
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30

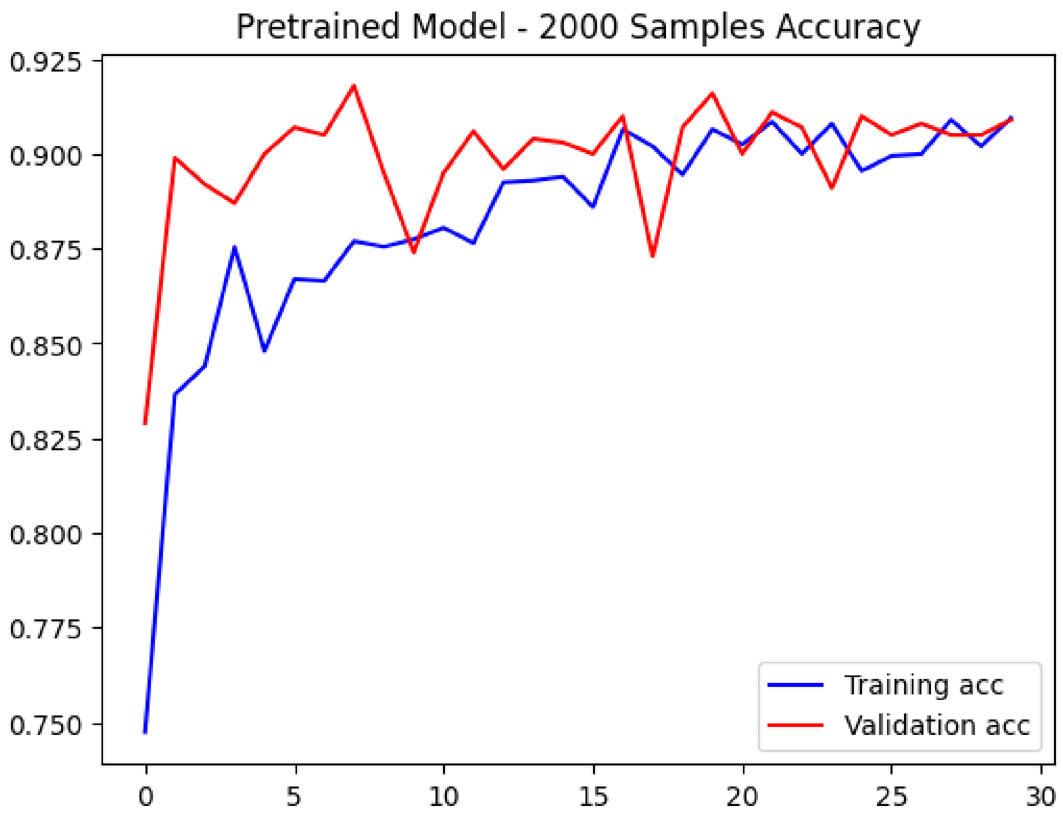

Model from Scratch - 2000 Samples Loss
Training loss
Validation loss
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
5
10
15
20
25
30

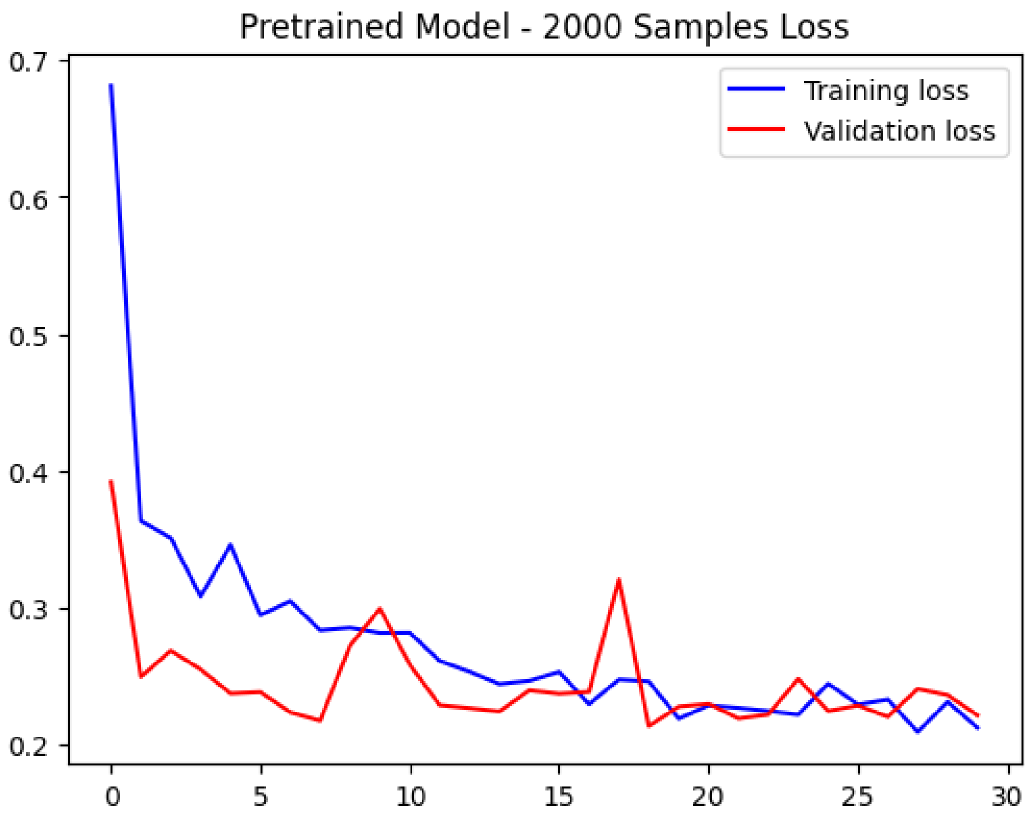

Pretrained Model - 1000 Samples Accuracy
0.70
0.75
0.80
0.85
0.90
0
5
10
15
20
25
30
Training acc
Validation acc


Pretrained Model - 1000 Samples Loss
Training loss
Validation loss
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25
30


Pretrained Model - 1500 Samples Accuracy
0.70
0.75
0.80
0.85
0.90
0
5
10
15
20
25
30
Training acc
Validation acc


Pretrained Model - 1500 Samples Loss
Training loss
Validation loss
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25
30

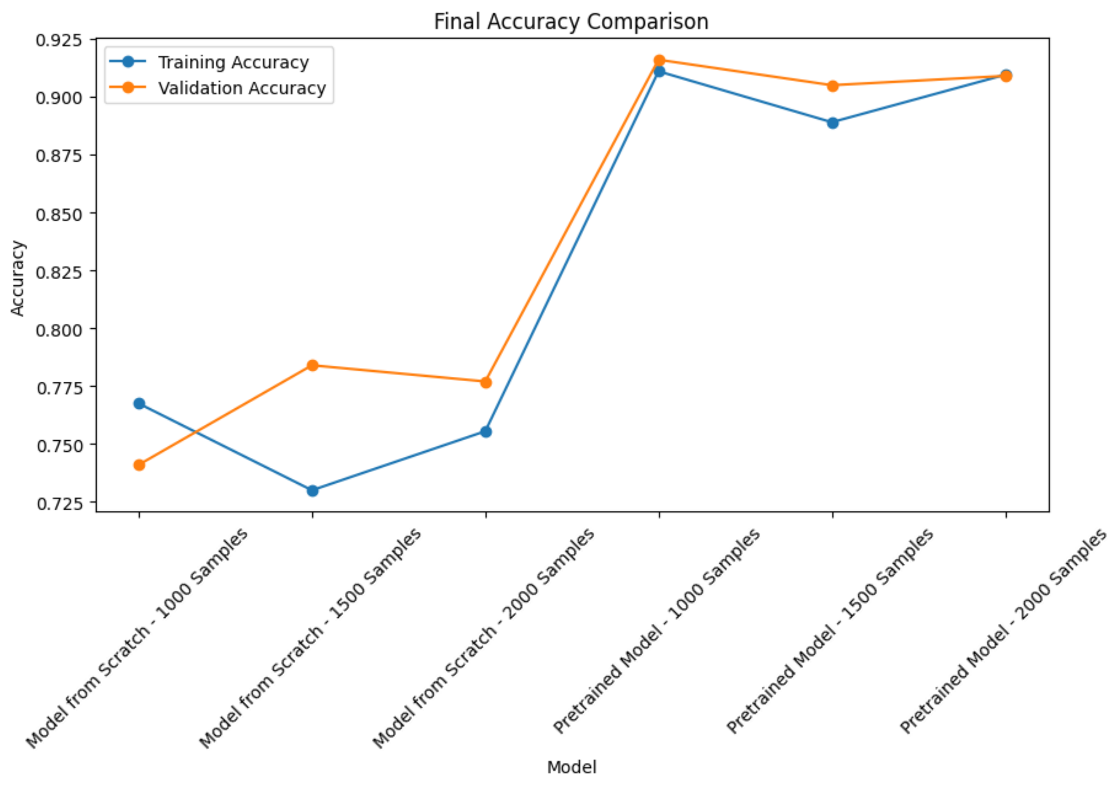

Pretrained Model - 2000 Samples Accuracy
0.925
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0
5
10
15
20
25
30
Training acc
Validation acc

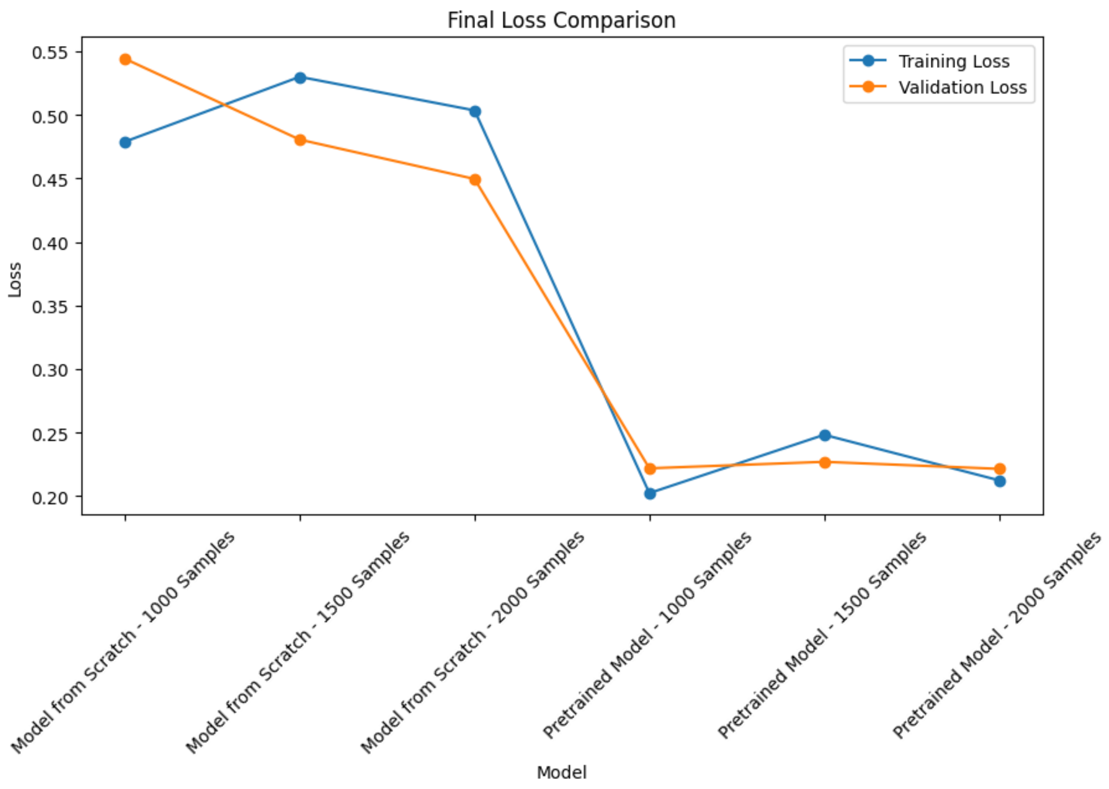


[37]:
```python
# Function to summarize final results
def aggregate_results(all_histories, classifier_model_image_labels):
    summary = {}
    for i, training_log in enumerate(all_histories):
        final_train_acc = training_log.history['accuracy'][-1] # Access history␣
↪directly
        final_val_acc = training_log.history['val_accuracy'][-1] # Access␣
↪history directly
        final_train_train_loss = training_log.history['loss'][-1] # Access␣
↪history directly
        final_validation_train_loss = training_log.history['val_loss'][-1] #␣
↪Access history directly
        summary[classifier_model_image_labels[i]] = {
            'Final Training Accuracy': final_train_acc,
            'Final Validation Accuracy': final_val_acc,
            'Final Training Loss': final_train_train_loss,
            'Final Validation Loss': final_validation_train_loss,
        }
    return summary
```

```
# Example of using the aggregate_results function
all_histories = [training_log_A, training_log_B, training_log_C,
             training_log_P1, training_log_P2, training_log_P3]
classifier_model_image_labels = [
    'Model from Scratch - 1000 Samples',
    'Model from Scratch - 1500 Samples',
    'Model from Scratch - 2000 Samples',
  'Pretrained Model - 1000 Samples',
    'Pretrained Model - 1500 Samples',
    'Pretrained Model - 2000 Samples'
]

# Get the summary of final values
final_scores = aggregate_results(all_histories, classifier_model_image_labels)␣
 ↪# Call the function to get the summary
```

[45]:
```
def compare_classifier_models(final_scores):
    classifier_model_image_labels = list(final_scores.keys())
    train_acc = [final_scores[image_label]['Final Training Accuracy'] for␣
 ↪image_label in classifier_model_image_labels]
    val_acc = [final_scores[image_label]['Final Validation Accuracy'] for␣
 ↪image_label in classifier_model_image_labels]
    train_train_loss = [final_scores[image_label]['Final Training Loss'] for␣
 ↪image_label in classifier_model_image_labels]
    validation_train_loss = [final_scores[image_label]['Final Validation Loss']␣
 ↪for image_label in classifier_model_image_labels]

    # Accuracy comparison plot
    plotlib.figure(figsize=(10, 5))
    # Changed 'image_label' to 'label'
    plotlib.plot(classifier_model_image_labels, train_acc, label='Training␣
 ↪Accuracy', marker='o')
    # Changed 'image_label' to 'label'
    plotlib.plot(classifier_model_image_labels, val_acc, label='Validation␣
 ↪Accuracy', marker='o')
    plotlib.xticks(rotation=45)
    plotlib.title('Final Accuracy Comparison')
    plotlib.xlabel('Model')  # Changed 'ximage_label' to 'xlabel'
    plotlib.ylabel('Accuracy')  # Changed 'yimage_label' to 'ylabel'
    plotlib.legend()
    plotlib.show()

    # Loss comparison plot
    plotlib.figure(figsize=(10, 5))
    # Changed 'image_label' to 'label'
    plotlib.plot(classifier_model_image_labels, train_train_loss,␣
↪label='Training Loss', marker='o')
    # Changed 'image_label' to 'label'
    plotlib.plot(classifier_model_image_labels, validation_train_loss,␣
↪label='Validation Loss', marker='o')
    plotlib.xticks(rotation=45)
    plotlib.title('Final Loss Comparison')
    plotlib.xlabel('Model')  # Changed 'ximage_label' to 'xlabel'
    plotlib.ylabel('Loss')  # Changed 'yimage_label' to 'ylabel'
    plotlib.legend()
    plotlib.show()

    # Plot the comparisons
compare_classifier_models(final_scores)
```




Final Loss Comparison
Training Loss
Validation Loss
Loss
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
Model from Scratch - 1000 Samples
Model from Scratch - 1500 Samples
Model from Scratch - 2000 Samples
Pretrained Model - 1000 Samples
Pretrained Model - 1500 Samples
Pretrained Model - 2000 Samples
Model