assignment-3

March 25, 2025

[2]:
```
from google.colab import drive
drive. mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[4]:
```
import zipfile
with zipfile.ZipFile("/content/drive/MyDrive/Colab Notebooks/cats_vs_dogs_small.
 ↪zip","r") as dataset_zip:
    dataset_zip.extractall("/content")
```

[5]:
```
import os
data_root = "/content/cats_vs_dogs_small"
print( "Contents of the base directory:", os.listdir(data_root))
```

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

[7]:
```
import os
import pathlib
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models, applications

# Establishing the dataset_results function's proper path
training_path = '/content/cats_vs_dogs_small/train'
validation_path = '/content/cats_vs_dogs_small/validation'
testing_path = '/content/cats_vs_dogs_small/test'
```

[8]:
```
# Loading the required datasets
ds_train = keras.preprocessing.image_dataset_from_directory(
    training_path,
    image_size=(180, 180),
    batch_size=32)

ds_val = keras.preprocessing.image_dataset_from_directory(
    validation_path,
```

```
    image_size=(180, 180),
    batch_size=32)

ds_test = keras.preprocessing.image_dataset_from_directory(
    testing_path,
    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.
```

[9]:
```
# Displaying the sample images
def show_images(dataset):
    plt.figure(figsize=(10, 10))
    for images, model_labels in dataset.take(1):
        for i in range(9):
            ax = plt.subplot(3, 3, i + 1)
            plt.imshow(images[i].numpy().astype("uint8"))
```

[10]:
```
# Displaying the sample images
def show_images(dataset):
    plt.figure(figsize=(10, 10))
    for images, model_labels in dataset.take(1):
        for i in range(9):
            ax = plt.subplot(3, 3, i + 1)
            plt.imshow(images[i].numpy().astype("uint8"))
            plt.title("Cat" if model_labels[i] == 0 else "Dog")
            plt.axis("off")
    plt.show()

# Call the function to display the images
show_images(ds_train)
```

[11]:
```python
# Function to create a CNN cnn_model from scratch
def build_custom_model():
    cnn_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_model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return cnn_model
```

[12]:
```
# Data augmentation and the preprocessing with sample size parameter
def prepare_data_flows(training_path, validation_path, 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(
        training_path,
        target_size=(180, 180),
        batch_size=batch_size,
        class_mode='binary'
    )

    val_generator = val_datagen.flow_from_directory(
        validation_path,
        target_size=(180, 180),
        batch_size=batch_size,
        class_mode='binary'
    )

    return gen_train, val_generator
```

[15]:
```
# Training the cnn_model
def fit_model(cnn_model, gen_train, gen_val, num_epochs=30):
    training_log = cnn_model.fit(gen_train,
                         validation_data=gen_val,
                                 epochs=num_epochs) # Changed 'num_epochs' to␣
 ↪'epochs'
    return training_log
```

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

model_A = build_custom_model()

history_A = fit_model(model_A, train_generator_1, validation_generator_1)
```

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

/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)
/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 382ms/step -
accuracy: 0.5011 - loss: 1.0270 - val_accuracy: 0.5690 - val_loss: 0.6897
Epoch 2/30
63/63             25s 387ms/step -
accuracy: 0.5660 - loss: 0.6886 - val_accuracy: 0.5850 - val_loss: 0.6806
Epoch 3/30
63/63             24s 383ms/step -
accuracy: 0.5731 - loss: 0.6644 - val_accuracy: 0.6040 - val_loss: 0.6601
Epoch 4/30
63/63             24s 381ms/step -
accuracy: 0.5835 - loss: 0.6674 - val_accuracy: 0.6420 - val_loss: 0.6328
Epoch 5/30
63/63             24s 382ms/step -
accuracy: 0.6074 - loss: 0.6659 - val_accuracy: 0.5650 - val_loss: 0.7205
Epoch 6/30
63/63             24s 377ms/step -
accuracy: 0.5596 - loss: 0.6888 - val_accuracy: 0.6040 - val_loss: 0.6609
Epoch 7/30
63/63             24s 380ms/step -
accuracy: 0.5702 - loss: 0.6822 - val_accuracy: 0.5760 - val_loss: 0.6650
Epoch 8/30
63/63             24s 378ms/step -
```

```
accuracy: 0.6231 - loss: 0.6651 - val_accuracy: 0.6580 - val_loss: 0.6500
Epoch 9/30
63/63             25s 388ms/step -
accuracy: 0.6469 - loss: 0.6440 - val_accuracy: 0.6630 - val_loss: 0.5963
Epoch 10/30
63/63             24s 378ms/step -
accuracy: 0.6579 - loss: 0.6155 - val_accuracy: 0.6140 - val_loss: 0.6338
Epoch 11/30
63/63             24s 383ms/step -
accuracy: 0.6420 - loss: 0.6276 - val_accuracy: 0.6680 - val_loss: 0.6006
Epoch 12/30
63/63             24s 381ms/step -
accuracy: 0.6805 - loss: 0.6047 - val_accuracy: 0.5760 - val_loss: 0.6361
Epoch 13/30
63/63             25s 387ms/step -
accuracy: 0.6532 - loss: 0.6181 - val_accuracy: 0.6620 - val_loss: 0.5958
Epoch 14/30
63/63             24s 383ms/step -
accuracy: 0.6670 - loss: 0.6160 - val_accuracy: 0.7060 - val_loss: 0.5571
Epoch 15/30
63/63             24s 381ms/step -
accuracy: 0.7141 - loss: 0.5791 - val_accuracy: 0.6870 - val_loss: 0.5880
Epoch 16/30
63/63             24s 382ms/step -
accuracy: 0.6643 - loss: 0.6086 - val_accuracy: 0.6990 - val_loss: 0.5674
Epoch 17/30
63/63             24s 384ms/step -
accuracy: 0.6782 - loss: 0.6015 - val_accuracy: 0.7040 - val_loss: 0.5499
Epoch 18/30
63/63             24s 380ms/step -
accuracy: 0.7219 - loss: 0.5545 - val_accuracy: 0.6940 - val_loss: 0.5451
Epoch 19/30
63/63             24s 373ms/step -
accuracy: 0.7123 - loss: 0.5580 - val_accuracy: 0.7500 - val_loss: 0.5205
Epoch 20/30
63/63             24s 373ms/step -
accuracy: 0.7263 - loss: 0.5490 - val_accuracy: 0.7090 - val_loss: 0.5308
Epoch 21/30
63/63             23s 366ms/step -
accuracy: 0.6894 - loss: 0.5812 - val_accuracy: 0.7380 - val_loss: 0.5156
Epoch 22/30
63/63             24s 373ms/step -
accuracy: 0.7170 - loss: 0.5477 - val_accuracy: 0.7140 - val_loss: 0.5328
Epoch 23/30
63/63             24s 372ms/step -
accuracy: 0.7264 - loss: 0.5448 - val_accuracy: 0.7300 - val_loss: 0.5269
Epoch 24/30
63/63             24s 372ms/step -
```

```
accuracy: 0.7318 - loss: 0.5386 - val_accuracy: 0.7540 - val_loss: 0.5028
Epoch 25/30
63/63             24s 373ms/step -
accuracy: 0.7412 - loss: 0.5353 - val_accuracy: 0.7430 - val_loss: 0.5104
Epoch 26/30
63/63             24s 371ms/step -
accuracy: 0.7385 - loss: 0.5416 - val_accuracy: 0.7460 - val_loss: 0.5010
Epoch 27/30
63/63             24s 373ms/step -
accuracy: 0.7353 - loss: 0.5303 - val_accuracy: 0.7600 - val_loss: 0.4972
Epoch 28/30
63/63             23s 370ms/step -
accuracy: 0.7303 - loss: 0.5249 - val_accuracy: 0.7610 - val_loss: 0.4908
Epoch 29/30
63/63             24s 373ms/step -
accuracy: 0.7399 - loss: 0.5085 - val_accuracy: 0.7520 - val_loss: 0.5058
Epoch 30/30
63/63             23s 368ms/step -
accuracy: 0.7614 - loss: 0.5101 - val_accuracy: 0.7660 - val_loss: 0.4832
```

[17]:
```
# Step 2: Increasing the training samples from 1000 to 1500
train_generator_2, validation_generator_2 = prepare_data_flows(training_path,␣
 ↪validation_path, batch_size=32, num_samples=1500)
model_B = build_custom_model()
history_B = fit_model(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             26s 377ms/step -
accuracy: 0.4971 - loss: 1.3384 - val_accuracy: 0.5490 - val_loss: 0.6928
Epoch 2/30
63/63             23s 370ms/step -
accuracy: 0.5192 - loss: 0.6931 - val_accuracy: 0.5000 - val_loss: 0.6944
Epoch 3/30
63/63             23s 368ms/step -
accuracy: 0.4954 - loss: 0.6981 - val_accuracy: 0.5000 - val_loss: 0.6930
Epoch 4/30
63/63             24s 371ms/step -
accuracy: 0.5059 - loss: 0.6931 - val_accuracy: 0.5400 - val_loss: 0.6916
Epoch 5/30
63/63             24s 376ms/step -
accuracy: 0.5491 - loss: 0.6970 - val_accuracy: 0.5000 - val_loss: 0.7021
Epoch 6/30
63/63             24s 373ms/step -
accuracy: 0.5034 - loss: 0.6903 - val_accuracy: 0.5490 - val_loss: 0.6802
Epoch 7/30
63/63             24s 377ms/step -
```

```
accuracy: 0.5339 - loss: 0.6839 - val_accuracy: 0.5240 - val_loss: 0.6858
Epoch 8/30
63/63            24s 371ms/step -
accuracy: 0.5495 - loss: 0.6855 - val_accuracy: 0.5600 - val_loss: 0.6769
Epoch 9/30
63/63            24s 371ms/step -
accuracy: 0.5603 - loss: 0.6829 - val_accuracy: 0.5560 - val_loss: 0.6774
Epoch 10/30
63/63            24s 372ms/step -
accuracy: 0.5649 - loss: 0.6762 - val_accuracy: 0.6220 - val_loss: 0.6402
Epoch 11/30
63/63            24s 373ms/step -
accuracy: 0.5759 - loss: 0.6647 - val_accuracy: 0.6250 - val_loss: 0.6294
Epoch 12/30
63/63            24s 373ms/step -
accuracy: 0.5957 - loss: 0.6574 - val_accuracy: 0.5860 - val_loss: 0.6608
Epoch 13/30
63/63            24s 371ms/step -
accuracy: 0.6097 - loss: 0.6464 - val_accuracy: 0.6650 - val_loss: 0.6145
Epoch 14/30
63/63            23s 368ms/step -
accuracy: 0.6202 - loss: 0.6446 - val_accuracy: 0.6510 - val_loss: 0.6309
Epoch 15/30
63/63            24s 372ms/step -
accuracy: 0.6298 - loss: 0.6422 - val_accuracy: 0.6730 - val_loss: 0.5974
Epoch 16/30
63/63            23s 369ms/step -
accuracy: 0.6785 - loss: 0.6099 - val_accuracy: 0.6520 - val_loss: 0.6014
Epoch 17/30
63/63            24s 371ms/step -
accuracy: 0.6706 - loss: 0.6203 - val_accuracy: 0.6930 - val_loss: 0.5746
Epoch 18/30
63/63            24s 370ms/step -
accuracy: 0.7024 - loss: 0.5779 - val_accuracy: 0.6850 - val_loss: 0.5846
Epoch 19/30
63/63            23s 369ms/step -
accuracy: 0.6816 - loss: 0.5833 - val_accuracy: 0.7220 - val_loss: 0.5585
Epoch 20/30
63/63            24s 371ms/step -
accuracy: 0.6658 - loss: 0.5844 - val_accuracy: 0.7300 - val_loss: 0.5389
Epoch 21/30
63/63            23s 364ms/step -
accuracy: 0.7171 - loss: 0.5563 - val_accuracy: 0.7560 - val_loss: 0.5245
Epoch 22/30
63/63            24s 374ms/step -
accuracy: 0.7106 - loss: 0.5490 - val_accuracy: 0.7430 - val_loss: 0.5208
Epoch 23/30
63/63            24s 373ms/step -
```

```
accuracy: 0.7479 - loss: 0.5138 - val_accuracy: 0.7410 - val_loss: 0.5369
Epoch 24/30
63/63             24s 376ms/step -
accuracy: 0.7285 - loss: 0.5322 - val_accuracy: 0.7360 - val_loss: 0.5342
Epoch 25/30
63/63             24s 375ms/step -
accuracy: 0.7157 - loss: 0.5375 - val_accuracy: 0.7150 - val_loss: 0.5134
Epoch 26/30
63/63             23s 370ms/step -
accuracy: 0.7349 - loss: 0.5289 - val_accuracy: 0.6980 - val_loss: 0.6232
Epoch 27/30
63/63             24s 374ms/step -
accuracy: 0.7437 - loss: 0.5360 - val_accuracy: 0.7540 - val_loss: 0.5000
Epoch 28/30
63/63             23s 369ms/step -
accuracy: 0.7656 - loss: 0.5047 - val_accuracy: 0.7530 - val_loss: 0.5151
Epoch 29/30
63/63             23s 366ms/step -
accuracy: 0.7644 - loss: 0.4801 - val_accuracy: 0.7710 - val_loss: 0.5059
Epoch 30/30
63/63             24s 374ms/step -
accuracy: 0.7466 - loss: 0.5087 - val_accuracy: 0.7540 - val_loss: 0.5228
```

[18]:
```
# Step 3: Now,using the total 2000 samples
train_generator_3, validation_generator_3 = prepare_data_flows(training_path,␣
 ↪validation_path, batch_size=32, num_samples=2000)
model_C = build_custom_model()
history_C = fit_model(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             26s 380ms/step -
accuracy: 0.5096 - loss: 0.9308 - val_accuracy: 0.4900 - val_loss: 0.6930
Epoch 2/30
63/63             24s 374ms/step -
accuracy: 0.5026 - loss: 0.6930 - val_accuracy: 0.5000 - val_loss: 0.6930
Epoch 3/30
63/63             23s 369ms/step -
accuracy: 0.4992 - loss: 0.6934 - val_accuracy: 0.5470 - val_loss: 0.6921
Epoch 4/30
63/63             24s 371ms/step -
accuracy: 0.5338 - loss: 0.6974 - val_accuracy: 0.5800 - val_loss: 0.6807
Epoch 5/30
63/63             24s 371ms/step -
accuracy: 0.5489 - loss: 0.6953 - val_accuracy: 0.5730 - val_loss: 0.6760
Epoch 6/30
63/63             23s 369ms/step -
```

```
accuracy: 0.5688 - loss: 0.6809 - val_accuracy: 0.6330 - val_loss: 0.6461
Epoch 7/30
63/63             24s 377ms/step -
accuracy: 0.6079 - loss: 0.6666 - val_accuracy: 0.5700 - val_loss: 0.6530
Epoch 8/30
63/63             24s 371ms/step -
accuracy: 0.5784 - loss: 0.6707 - val_accuracy: 0.6390 - val_loss: 0.6346
Epoch 9/30
63/63             24s 374ms/step -
accuracy: 0.6240 - loss: 0.6791 - val_accuracy: 0.6630 - val_loss: 0.6229
Epoch 10/30
63/63             24s 374ms/step -
accuracy: 0.6239 - loss: 0.6561 - val_accuracy: 0.6020 - val_loss: 0.6469
Epoch 11/30
63/63             23s 370ms/step -
accuracy: 0.6111 - loss: 0.6667 - val_accuracy: 0.6860 - val_loss: 0.6118
Epoch 12/30
63/63             23s 369ms/step -
accuracy: 0.6447 - loss: 0.6460 - val_accuracy: 0.6750 - val_loss: 0.6132
Epoch 13/30
63/63             23s 369ms/step -
accuracy: 0.6434 - loss: 0.6346 - val_accuracy: 0.6800 - val_loss: 0.5932
Epoch 14/30
63/63             23s 370ms/step -
accuracy: 0.6477 - loss: 0.6319 - val_accuracy: 0.6810 - val_loss: 0.5914
Epoch 15/30
63/63             24s 374ms/step -
accuracy: 0.6715 - loss: 0.6168 - val_accuracy: 0.6910 - val_loss: 0.5791
Epoch 16/30
63/63             24s 377ms/step -
accuracy: 0.6617 - loss: 0.6135 - val_accuracy: 0.7100 - val_loss: 0.5654
Epoch 17/30
63/63             24s 380ms/step -
accuracy: 0.6790 - loss: 0.5884 - val_accuracy: 0.7090 - val_loss: 0.5585
Epoch 18/30
63/63             24s 371ms/step -
accuracy: 0.7200 - loss: 0.5752 - val_accuracy: 0.6640 - val_loss: 0.6017
Epoch 19/30
63/63             24s 374ms/step -
accuracy: 0.6600 - loss: 0.6131 - val_accuracy: 0.7200 - val_loss: 0.5589
Epoch 20/30
63/63             23s 367ms/step -
accuracy: 0.7117 - loss: 0.5627 - val_accuracy: 0.6910 - val_loss: 0.5829
Epoch 21/30
63/63             24s 373ms/step -
accuracy: 0.6805 - loss: 0.5906 - val_accuracy: 0.7380 - val_loss: 0.5426
Epoch 22/30
63/63             24s 378ms/step -
```

```
accuracy: 0.6863 - loss: 0.5717 - val_accuracy: 0.6820 - val_loss: 0.5807
Epoch 23/30
63/63             24s 380ms/step -
accuracy: 0.7210 - loss: 0.5521 - val_accuracy: 0.7210 - val_loss: 0.5317
Epoch 24/30
63/63             24s 373ms/step -
accuracy: 0.7278 - loss: 0.5440 - val_accuracy: 0.7050 - val_loss: 0.5499
Epoch 25/30
63/63             24s 374ms/step -
accuracy: 0.7162 - loss: 0.5495 - val_accuracy: 0.7480 - val_loss: 0.5286
Epoch 26/30
63/63             24s 373ms/step -
accuracy: 0.7069 - loss: 0.5583 - val_accuracy: 0.6820 - val_loss: 0.5762
Epoch 27/30
63/63             24s 379ms/step -
accuracy: 0.7217 - loss: 0.5492 - val_accuracy: 0.7240 - val_loss: 0.5320
Epoch 28/30
63/63             23s 368ms/step -
accuracy: 0.7292 - loss: 0.5411 - val_accuracy: 0.7480 - val_loss: 0.5171
Epoch 29/30
63/63             24s 372ms/step -
accuracy: 0.7512 - loss: 0.5164 - val_accuracy: 0.7520 - val_loss: 0.5158
Epoch 30/30
63/63             24s 372ms/step -
accuracy: 0.7450 - loss: 0.5212 - val_accuracy: 0.7050 - val_loss: 0.5455
```

[19]:
```python
# Step 4: Now use a pretrained cnn_model (e.g., VGG16)
def build_pretrained_model():
    base_model = applications.VGG16(include_top=False, weights='imagenet',␣
 ↪input_shape=(180, 180, 3))
    base_model.trainable = False  # Freeze the base cnn_model

    cnn_model = models.Sequential([
        base_model,
        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.Dense(1, activation='sigmoid')
    ])

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

[20]:
```python
# Repeating the Steps 1 to 3 with the pretrained cnn_model
model_P1 = build_pretrained_model()
history_P1 = fit_model(model_P1, train_generator_1, validation_generator_1)
```

```
model_P2 = build_pretrained_model()
history_P2 = fit_model(model_P2, train_generator_2, validation_generator_2)

model_P3 = build_pretrained_model()
history_P3 = fit_model(model_P3, train_generator_3, validation_generator_3)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256            0s
0us/step
Epoch 1/30
63/63             43s 666ms/step -
accuracy: 0.6499 - loss: 1.3855 - val_accuracy: 0.8680 - val_loss: 0.3160
Epoch 2/30
63/63             41s 656ms/step -
accuracy: 0.8329 - loss: 0.3674 - val_accuracy: 0.8950 - val_loss: 0.2389
Epoch 3/30
63/63             41s 652ms/step -
accuracy: 0.8352 - loss: 0.3720 - val_accuracy: 0.9040 - val_loss: 0.2244
Epoch 4/30
63/63             41s 650ms/step -
accuracy: 0.8678 - loss: 0.2976 - val_accuracy: 0.8240 - val_loss: 0.4290
Epoch 5/30
63/63             42s 661ms/step -
accuracy: 0.8318 - loss: 0.3490 - val_accuracy: 0.8970 - val_loss: 0.2562
Epoch 6/30
63/63             41s 656ms/step -
accuracy: 0.8590 - loss: 0.3156 - val_accuracy: 0.8870 - val_loss: 0.2724
Epoch 7/30
63/63             42s 662ms/step -
accuracy: 0.8495 - loss: 0.3382 - val_accuracy: 0.9140 - val_loss: 0.2175
Epoch 8/30
63/63             41s 656ms/step -
accuracy: 0.8880 - loss: 0.2587 - val_accuracy: 0.9030 - val_loss: 0.2215
Epoch 9/30
63/63             42s 660ms/step -
accuracy: 0.8707 - loss: 0.2933 - val_accuracy: 0.9130 - val_loss: 0.2255
Epoch 10/30
63/63             42s 659ms/step -
accuracy: 0.8833 - loss: 0.2754 - val_accuracy: 0.9140 - val_loss: 0.2097
Epoch 11/30
63/63             41s 656ms/step -
accuracy: 0.8918 - loss: 0.2376 - val_accuracy: 0.9070 - val_loss: 0.2248
Epoch 12/30
63/63             42s 659ms/step -
accuracy: 0.8924 - loss: 0.2464 - val_accuracy: 0.9170 - val_loss: 0.2108
Epoch 13/30
```

```
63/63            42s 664ms/step -
accuracy: 0.8981 - loss: 0.2392 - val_accuracy: 0.9080 - val_loss: 0.2148
Epoch 14/30
63/63            42s 659ms/step -
accuracy: 0.9032 - loss: 0.2388 - val_accuracy: 0.8990 - val_loss: 0.2261
Epoch 15/30
63/63            41s 651ms/step -
accuracy: 0.9053 - loss: 0.2294 - val_accuracy: 0.9130 - val_loss: 0.2070
Epoch 16/30
63/63            41s 656ms/step -
accuracy: 0.9036 - loss: 0.2299 - val_accuracy: 0.9130 - val_loss: 0.2218
Epoch 17/30
63/63            42s 660ms/step -
accuracy: 0.8985 - loss: 0.2327 - val_accuracy: 0.9050 - val_loss: 0.2212
Epoch 18/30
63/63            42s 659ms/step -
accuracy: 0.9016 - loss: 0.2310 - val_accuracy: 0.9080 - val_loss: 0.2210
Epoch 19/30
63/63            42s 666ms/step -
accuracy: 0.8924 - loss: 0.2413 - val_accuracy: 0.9120 - val_loss: 0.2180
Epoch 20/30
63/63            41s 657ms/step -
accuracy: 0.8973 - loss: 0.2203 - val_accuracy: 0.9120 - val_loss: 0.2227
Epoch 21/30
63/63            42s 659ms/step -
accuracy: 0.9160 - loss: 0.2182 - val_accuracy: 0.8960 - val_loss: 0.2479
Epoch 22/30
63/63            41s 657ms/step -
accuracy: 0.9113 - loss: 0.2132 - val_accuracy: 0.9150 - val_loss: 0.2167
Epoch 23/30
63/63            41s 658ms/step -
accuracy: 0.8979 - loss: 0.2255 - val_accuracy: 0.9100 - val_loss: 0.2192
Epoch 24/30
63/63            41s 657ms/step -
accuracy: 0.9009 - loss: 0.2232 - val_accuracy: 0.9100 - val_loss: 0.2303
Epoch 25/30
63/63            42s 662ms/step -
accuracy: 0.9105 - loss: 0.2105 - val_accuracy: 0.9100 - val_loss: 0.2289
Epoch 26/30
63/63            42s 660ms/step -
accuracy: 0.8970 - loss: 0.2362 - val_accuracy: 0.9140 - val_loss: 0.2351
Epoch 27/30
63/63            42s 663ms/step -
accuracy: 0.9119 - loss: 0.2175 - val_accuracy: 0.9150 - val_loss: 0.2210
Epoch 28/30
63/63            42s 660ms/step -
accuracy: 0.9035 - loss: 0.2131 - val_accuracy: 0.9170 - val_loss: 0.2190
Epoch 29/30
```

```
63/63            42s 663ms/step -
accuracy: 0.9186 - loss: 0.2014 - val_accuracy: 0.9150 - val_loss: 0.2162
Epoch 30/30
63/63            42s 662ms/step -
accuracy: 0.9240 - loss: 0.1893 - val_accuracy: 0.8930 - val_loss: 0.2519
Epoch 1/30
63/63            43s 668ms/step -
accuracy: 0.7152 - loss: 0.8685 - val_accuracy: 0.8490 - val_loss: 0.3769
Epoch 2/30
63/63            41s 657ms/step -
accuracy: 0.8387 - loss: 0.3838 - val_accuracy: 0.8910 - val_loss: 0.2675
Epoch 3/30
63/63            42s 664ms/step -
accuracy: 0.8425 - loss: 0.3442 - val_accuracy: 0.9090 - val_loss: 0.2361
Epoch 4/30
63/63            41s 658ms/step -
accuracy: 0.8591 - loss: 0.3109 - val_accuracy: 0.8940 - val_loss: 0.2623
Epoch 5/30
63/63            41s 655ms/step -
accuracy: 0.8749 - loss: 0.3123 - val_accuracy: 0.8710 - val_loss: 0.3041
Epoch 6/30
63/63            41s 656ms/step -
accuracy: 0.8707 - loss: 0.2863 - val_accuracy: 0.9100 - val_loss: 0.2189
Epoch 7/30
63/63            41s 658ms/step -
accuracy: 0.8818 - loss: 0.2803 - val_accuracy: 0.9120 - val_loss: 0.2201
Epoch 8/30
63/63            42s 659ms/step -
accuracy: 0.8927 - loss: 0.2613 - val_accuracy: 0.9130 - val_loss: 0.2229
Epoch 9/30
63/63            41s 656ms/step -
accuracy: 0.8698 - loss: 0.2896 - val_accuracy: 0.8980 - val_loss: 0.2388
Epoch 10/30
63/63            41s 655ms/step -
accuracy: 0.8789 - loss: 0.2690 - val_accuracy: 0.9050 - val_loss: 0.2351
Epoch 11/30
63/63            41s 654ms/step -
accuracy: 0.8835 - loss: 0.2859 - val_accuracy: 0.9090 - val_loss: 0.2308
Epoch 12/30
63/63            41s 652ms/step -
accuracy: 0.9000 - loss: 0.2371 - val_accuracy: 0.8910 - val_loss: 0.2573
Epoch 13/30
63/63            41s 653ms/step -
accuracy: 0.8754 - loss: 0.2796 - val_accuracy: 0.9070 - val_loss: 0.2136
Epoch 14/30
63/63            41s 654ms/step -
accuracy: 0.8987 - loss: 0.2410 - val_accuracy: 0.9170 - val_loss: 0.2220
Epoch 15/30
```

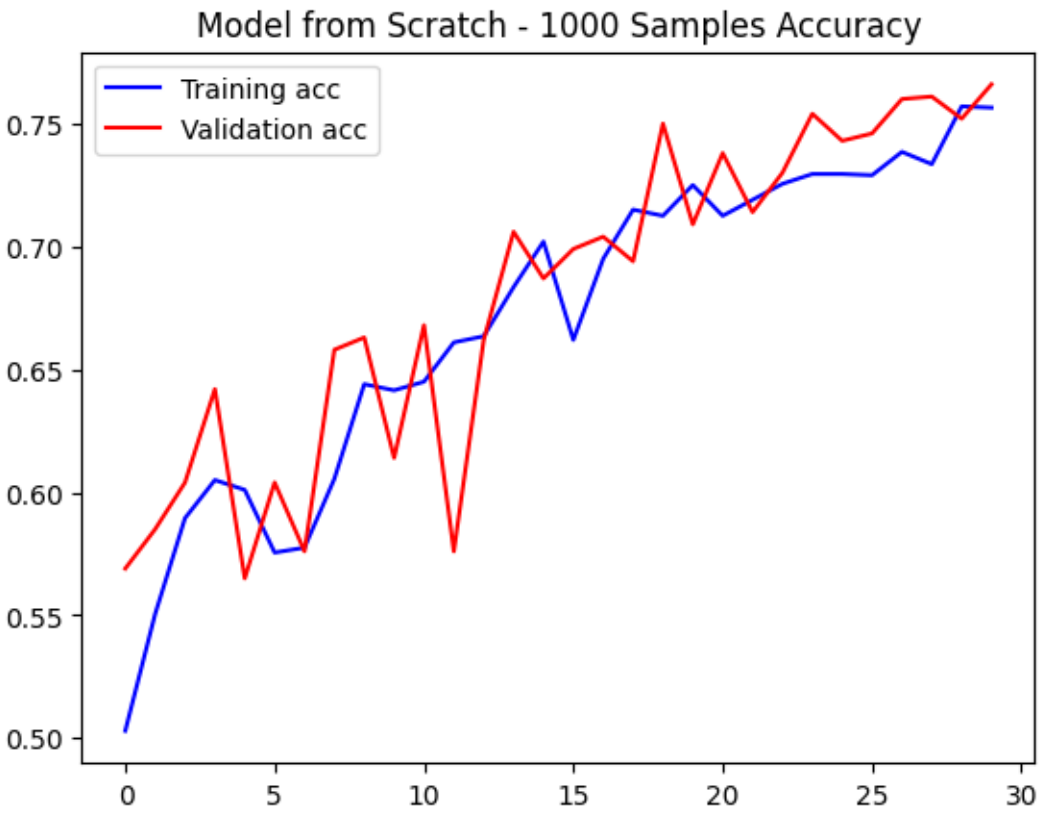
```
63/63            41s 654ms/step -
accuracy: 0.9012 - loss: 0.2430 - val_accuracy: 0.9140 - val_loss: 0.2139
Epoch 16/30
63/63            41s 651ms/step -
accuracy: 0.9026 - loss: 0.2566 - val_accuracy: 0.9150 - val_loss: 0.2268
Epoch 17/30
63/63            41s 652ms/step -
accuracy: 0.9014 - loss: 0.2481 - val_accuracy: 0.9070 - val_loss: 0.2195
Epoch 18/30
63/63            41s 653ms/step -
accuracy: 0.8969 - loss: 0.2345 - val_accuracy: 0.8600 - val_loss: 0.3280
Epoch 19/30
63/63            41s 655ms/step -
accuracy: 0.8771 - loss: 0.2706 - val_accuracy: 0.9020 - val_loss: 0.2395
Epoch 20/30
63/63            41s 658ms/step -
accuracy: 0.9035 - loss: 0.2306 - val_accuracy: 0.9140 - val_loss: 0.2198
Epoch 21/30
63/63            41s 653ms/step -
accuracy: 0.8996 - loss: 0.2280 - val_accuracy: 0.9040 - val_loss: 0.2304
Epoch 22/30
63/63            41s 657ms/step -
accuracy: 0.9016 - loss: 0.2245 - val_accuracy: 0.9050 - val_loss: 0.2457
Epoch 23/30
63/63            41s 656ms/step -
accuracy: 0.8944 - loss: 0.2526 - val_accuracy: 0.9100 - val_loss: 0.2273
Epoch 24/30
63/63            41s 655ms/step -
accuracy: 0.9167 - loss: 0.2189 - val_accuracy: 0.9090 - val_loss: 0.2326
Epoch 25/30
63/63            41s 652ms/step -
accuracy: 0.9082 - loss: 0.2106 - val_accuracy: 0.9110 - val_loss: 0.2070
Epoch 26/30
63/63            41s 653ms/step -
accuracy: 0.9014 - loss: 0.2291 - val_accuracy: 0.9040 - val_loss: 0.2419
Epoch 27/30
63/63            41s 647ms/step -
accuracy: 0.9028 - loss: 0.2273 - val_accuracy: 0.8960 - val_loss: 0.2459
Epoch 28/30
63/63            41s 657ms/step -
accuracy: 0.9033 - loss: 0.2270 - val_accuracy: 0.9190 - val_loss: 0.2161
Epoch 29/30
63/63            41s 651ms/step -
accuracy: 0.9110 - loss: 0.2005 - val_accuracy: 0.9050 - val_loss: 0.2311
Epoch 30/30
63/63            41s 657ms/step -
accuracy: 0.9044 - loss: 0.2150 - val_accuracy: 0.9100 - val_loss: 0.2272
Epoch 1/30
```

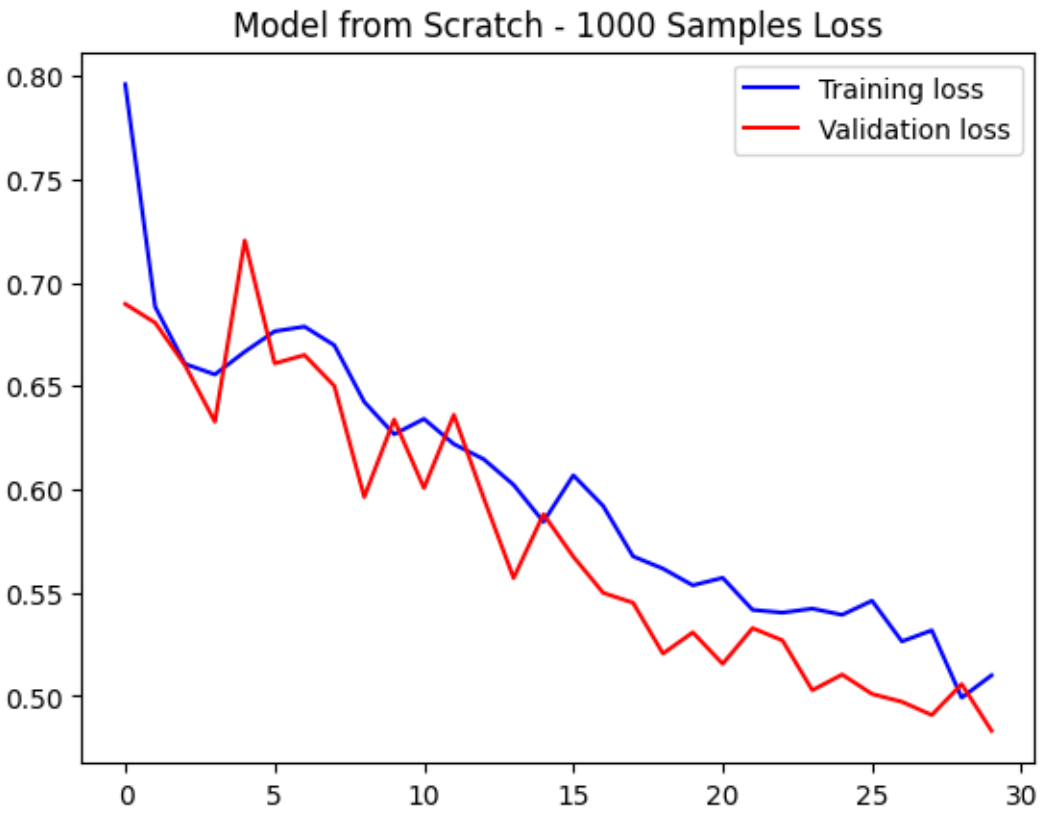
```
63/63            43s 662ms/step -
accuracy: 0.5993 - loss: 1.5840 - val_accuracy: 0.8670 - val_loss: 0.3195
Epoch 2/30
63/63            42s 660ms/step -
accuracy: 0.8262 - loss: 0.3867 - val_accuracy: 0.8920 - val_loss: 0.2745
Epoch 3/30
63/63            41s 658ms/step -
accuracy: 0.8631 - loss: 0.3231 - val_accuracy: 0.8970 - val_loss: 0.2467
Epoch 4/30
63/63            41s 655ms/step -
accuracy: 0.8494 - loss: 0.3287 - val_accuracy: 0.9050 - val_loss: 0.2383
Epoch 5/30
63/63            41s 657ms/step -
accuracy: 0.8636 - loss: 0.3041 - val_accuracy: 0.8990 - val_loss: 0.2559
Epoch 6/30
63/63            41s 657ms/step -
accuracy: 0.8744 - loss: 0.2753 - val_accuracy: 0.8920 - val_loss: 0.2470
Epoch 7/30
63/63            42s 660ms/step -
accuracy: 0.8681 - loss: 0.2829 - val_accuracy: 0.8960 - val_loss: 0.2181
Epoch 8/30
63/63            42s 661ms/step -
accuracy: 0.8993 - loss: 0.2550 - val_accuracy: 0.8990 - val_loss: 0.2312
Epoch 9/30
63/63            42s 663ms/step -
accuracy: 0.8825 - loss: 0.2658 - val_accuracy: 0.9040 - val_loss: 0.2324
Epoch 10/30
63/63            41s 652ms/step -
accuracy: 0.8770 - loss: 0.2697 - val_accuracy: 0.9090 - val_loss: 0.2175
Epoch 11/30
63/63            41s 656ms/step -
accuracy: 0.8831 - loss: 0.2504 - val_accuracy: 0.9130 - val_loss: 0.2260
Epoch 12/30
63/63            41s 656ms/step -
accuracy: 0.8785 - loss: 0.2741 - val_accuracy: 0.9040 - val_loss: 0.2125
Epoch 13/30
63/63            41s 654ms/step -
accuracy: 0.8894 - loss: 0.2552 - val_accuracy: 0.9020 - val_loss: 0.2274
Epoch 14/30
63/63            41s 651ms/step -
accuracy: 0.8964 - loss: 0.2325 - val_accuracy: 0.9110 - val_loss: 0.2126
Epoch 15/30
63/63            42s 665ms/step -
accuracy: 0.8954 - loss: 0.2526 - val_accuracy: 0.9040 - val_loss: 0.2272
Epoch 16/30
63/63            42s 665ms/step -
accuracy: 0.8969 - loss: 0.2388 - val_accuracy: 0.9140 - val_loss: 0.2202
Epoch 17/30
```

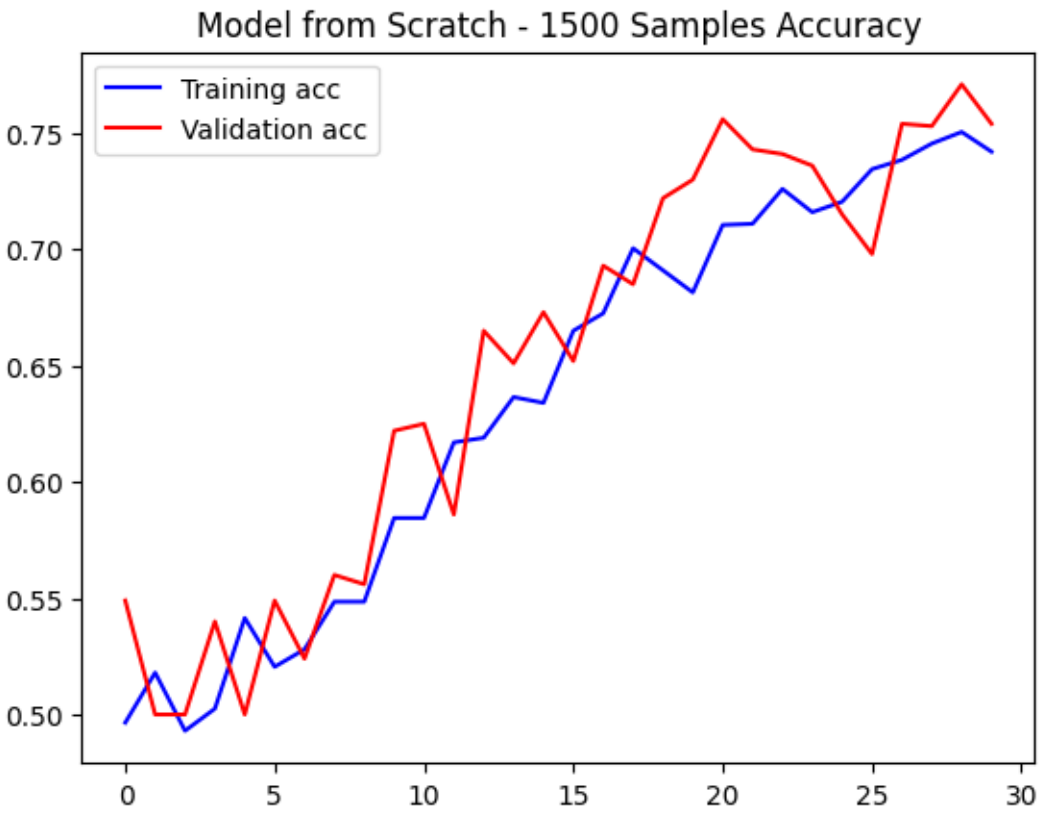
```
63/63            41s 656ms/step -
accuracy: 0.9027 - loss: 0.2496 - val_accuracy: 0.9090 - val_loss: 0.2122
Epoch 18/30
63/63            41s 657ms/step -
accuracy: 0.9026 - loss: 0.2255 - val_accuracy: 0.8750 - val_loss: 0.2925
Epoch 19/30
63/63            41s 655ms/step -
accuracy: 0.8817 - loss: 0.2775 - val_accuracy: 0.9010 - val_loss: 0.2219
Epoch 20/30
63/63            42s 662ms/step -
accuracy: 0.9131 - loss: 0.2193 - val_accuracy: 0.9010 - val_loss: 0.2484
Epoch 21/30
63/63            42s 658ms/step -
accuracy: 0.8930 - loss: 0.2399 - val_accuracy: 0.9170 - val_loss: 0.2211
Epoch 22/30
63/63            42s 662ms/step -
accuracy: 0.9068 - loss: 0.2275 - val_accuracy: 0.9120 - val_loss: 0.2204
Epoch 23/30
63/63            41s 655ms/step -
accuracy: 0.9023 - loss: 0.2384 - val_accuracy: 0.9010 - val_loss: 0.2319
Epoch 24/30
63/63            42s 659ms/step -
accuracy: 0.8868 - loss: 0.2439 - val_accuracy: 0.9100 - val_loss: 0.2171
Epoch 25/30
63/63            42s 660ms/step -
accuracy: 0.8992 - loss: 0.2355 - val_accuracy: 0.9080 - val_loss: 0.2204
Epoch 26/30
63/63            42s 660ms/step -
accuracy: 0.9178 - loss: 0.2058 - val_accuracy: 0.9110 - val_loss: 0.2123
Epoch 27/30
63/63            42s 663ms/step -
accuracy: 0.9101 - loss: 0.2158 - val_accuracy: 0.9080 - val_loss: 0.2216
Epoch 28/30
63/63            41s 655ms/step -
accuracy: 0.9071 - loss: 0.2296 - val_accuracy: 0.9100 - val_loss: 0.2214
Epoch 29/30
63/63            42s 662ms/step -
accuracy: 0.9122 - loss: 0.2147 - val_accuracy: 0.9120 - val_loss: 0.2118
Epoch 30/30
63/63            41s 655ms/step -
accuracy: 0.9033 - loss: 0.2181 - val_accuracy: 0.9080 - val_loss: 0.2281
```

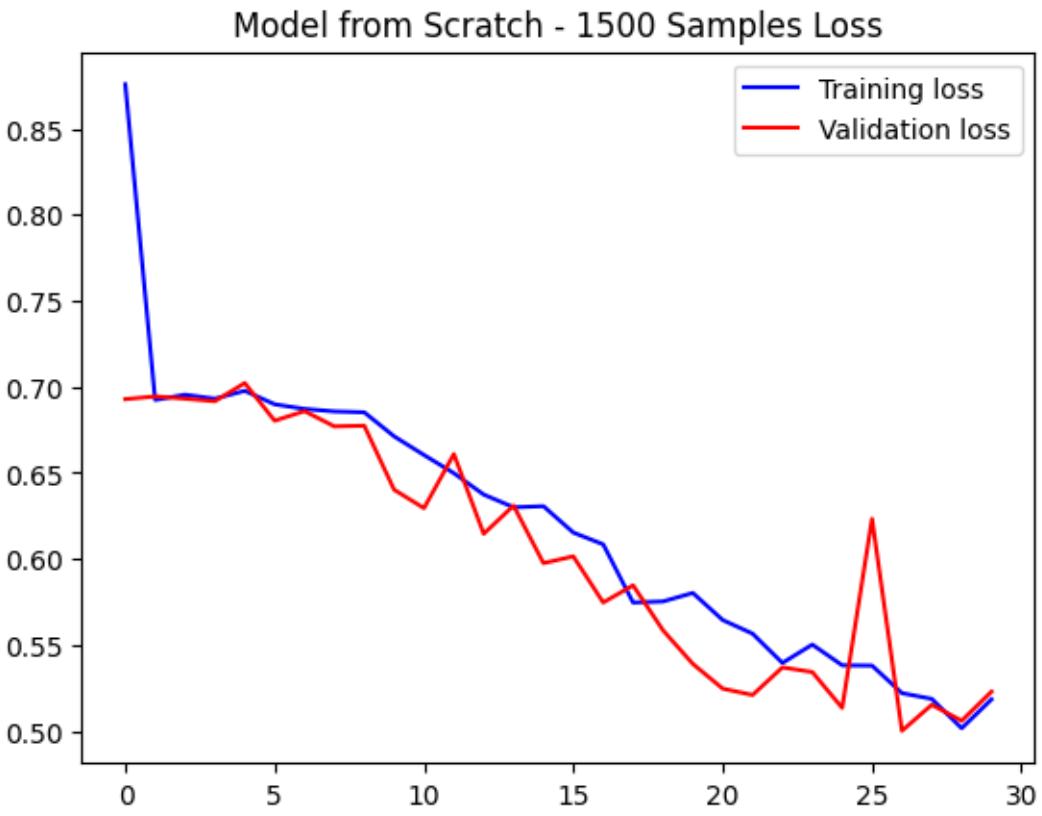
[28]:
```python
# Performance visualization function
def visualize_performance(training_log, title):
    acc = training_log.history['accuracy'] # Changed from training_log.
 ↪training_log['accuracy'] to training_log.history['accuracy']
```

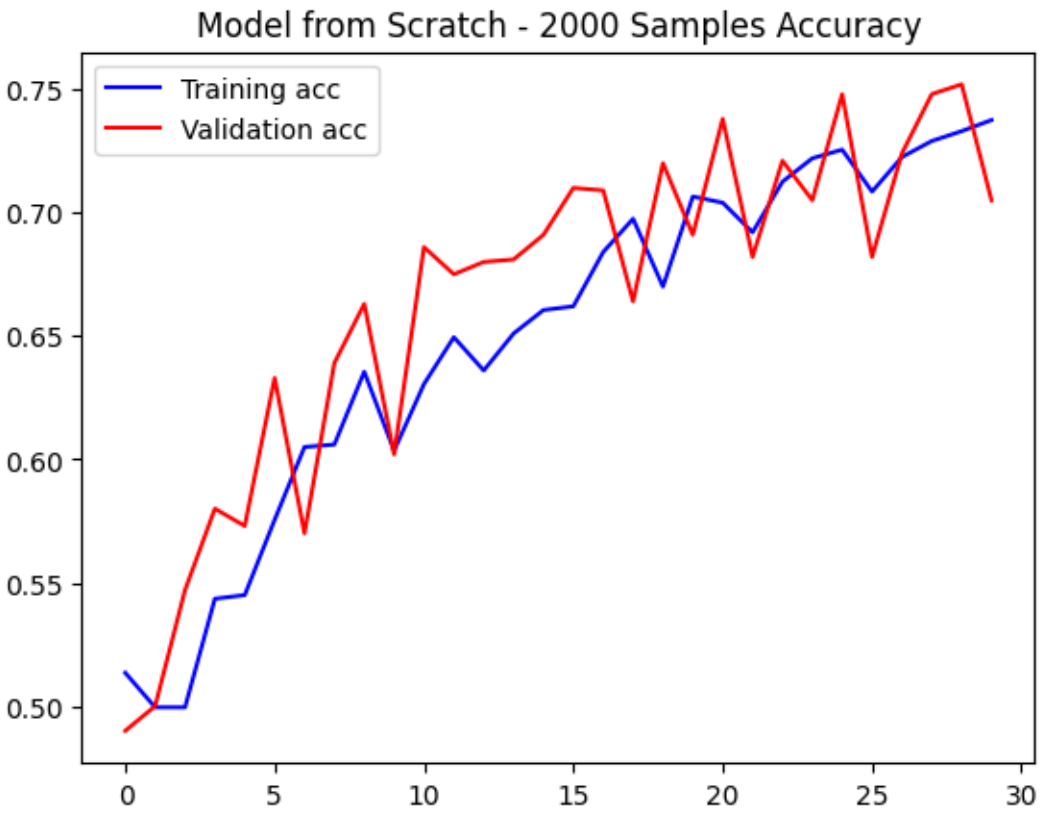
```
    val_acc = training_log.history['val_accuracy'] # Changed from training_log.
↪training_log['val_accuracy'] to training_log.history['val_accuracy']
    loss = training_log.history['loss'] # Changed from training_log.
↪training_log['loss'] to training_log.history['loss']
    val_loss = training_log.history['val_loss'] # Changed from training_log.
↪training_log['val_loss'] to training_log.history['val_loss']

    num_epochs = range(len(acc))

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

    plt.figure()
    plt.plot(num_epochs, loss, 'b', label='Training loss')
    plt.plot(num_epochs, val_loss, 'r', label='Validation loss')
    plt.title(title + ' Loss')
    plt.legend()
    plt.show()

# Plot for the performance of each cnn_model
visualize_performance(history_A, 'Model from Scratch - 1000 Samples')
visualize_performance(history_B, 'Model from Scratch - 1500 Samples')
visualize_performance(history_C, 'Model from Scratch - 2000 Samples')
visualize_performance(history_P1, 'Pretrained Model - 1000 Samples')
visualize_performance(history_P2, 'Pretrained Model - 1500 Samples')
visualize_performance(history_P3, 'Pretrained Model - 2000 Samples')
```

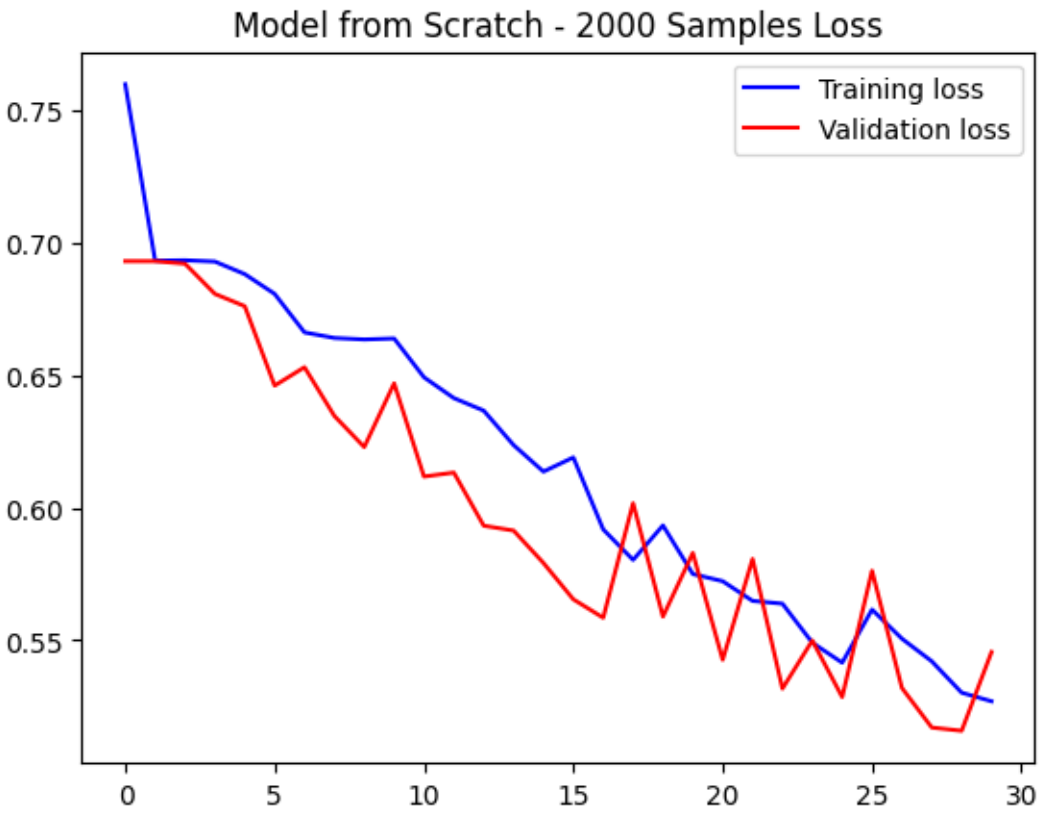

Model from Scratch - 1000 Samples Accuracy
Training acc
Validation acc
0.75
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30

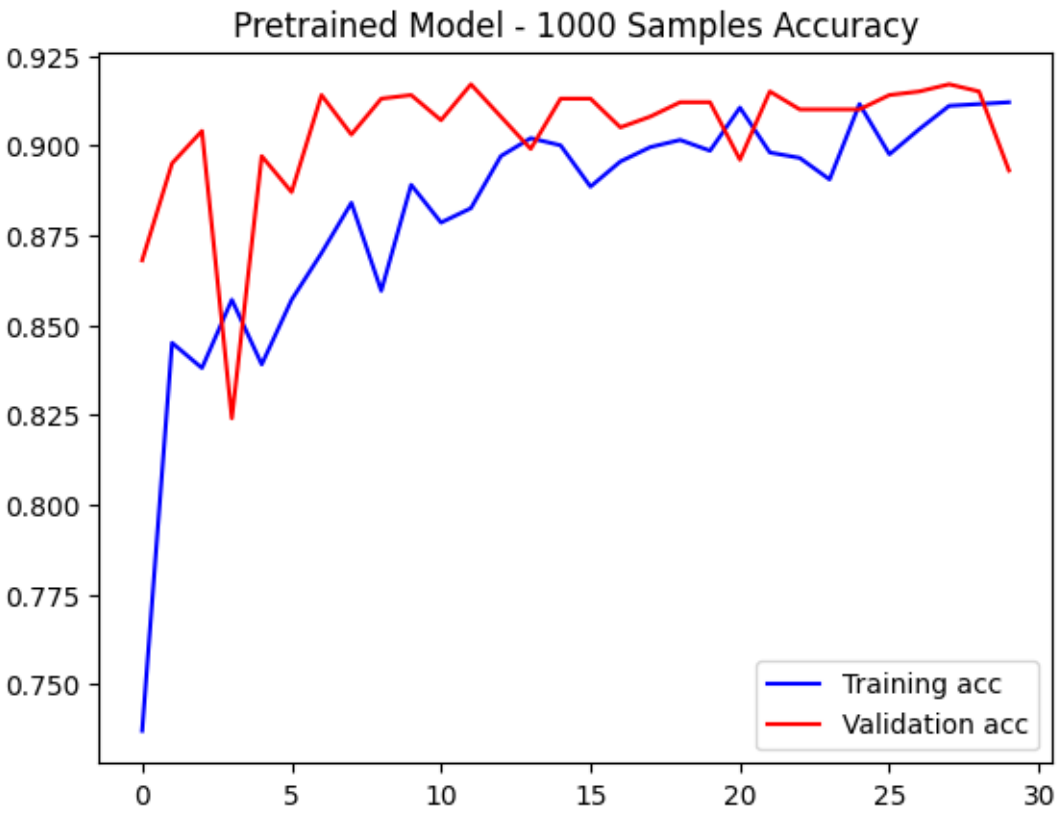

Model from Scratch - 1000 Samples Loss
Training loss
Validation loss
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30

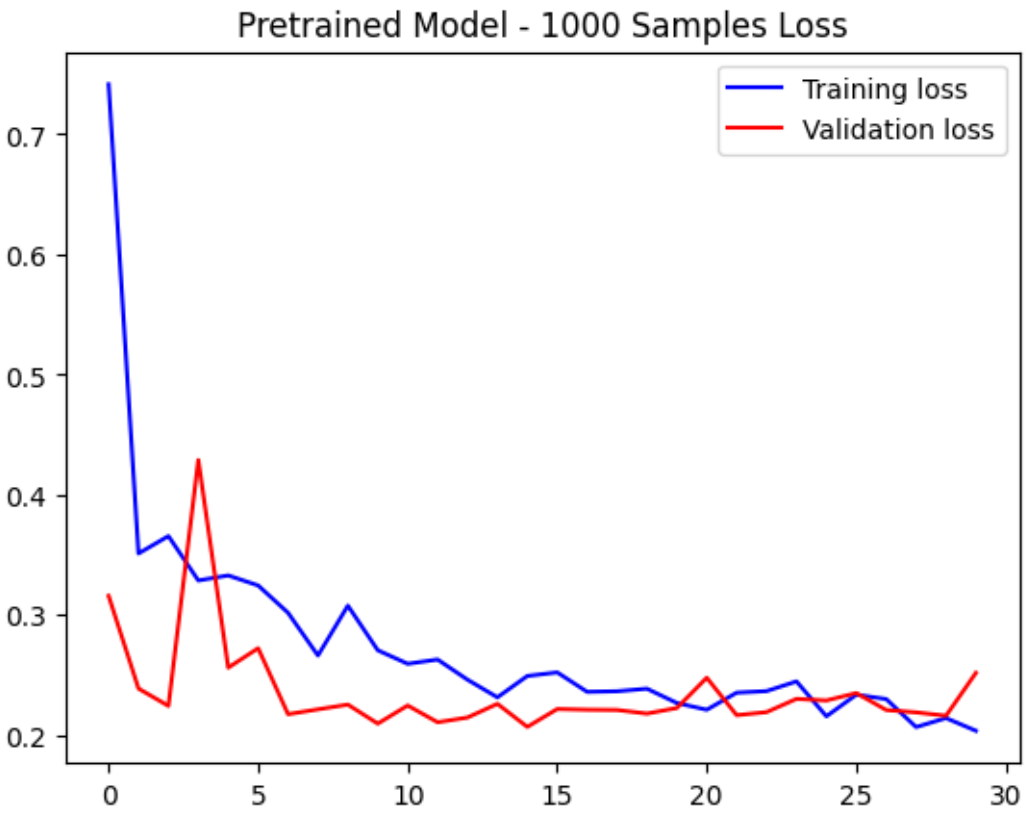

Model from Scratch - 1500 Samples Accuracy
Training acc
Validation acc
0.75
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30

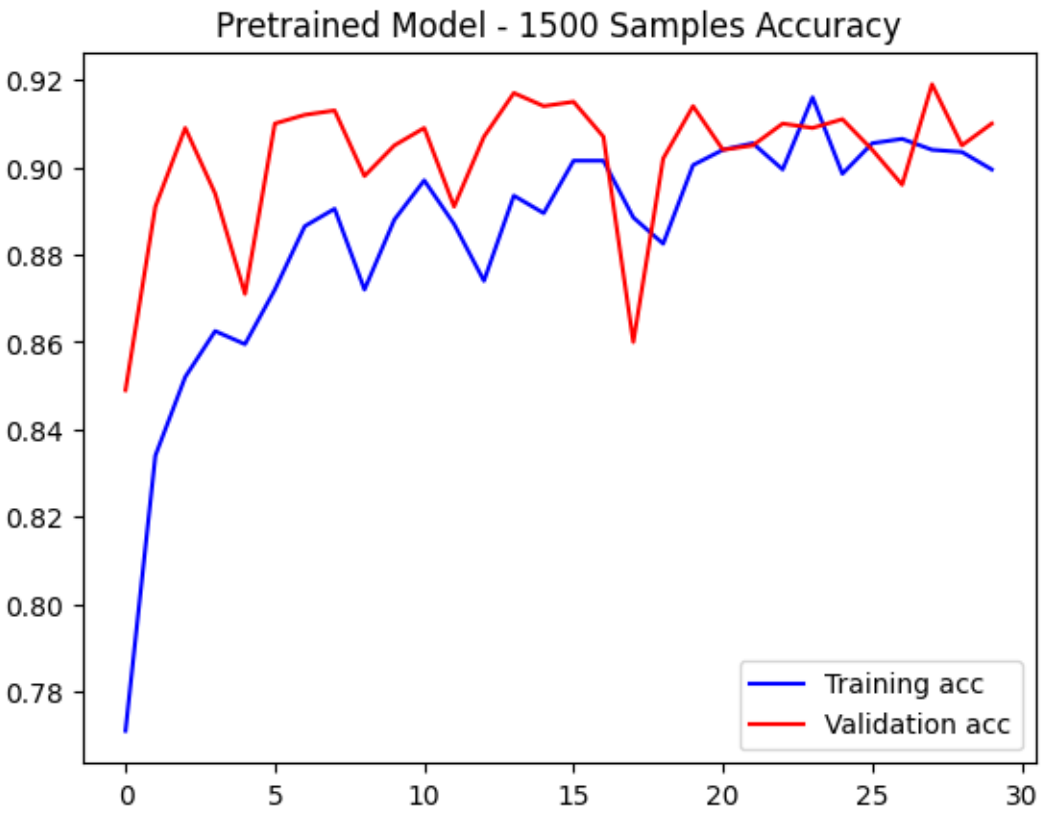

Model from Scratch - 1500 Samples Loss
Training loss
Validation loss
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30

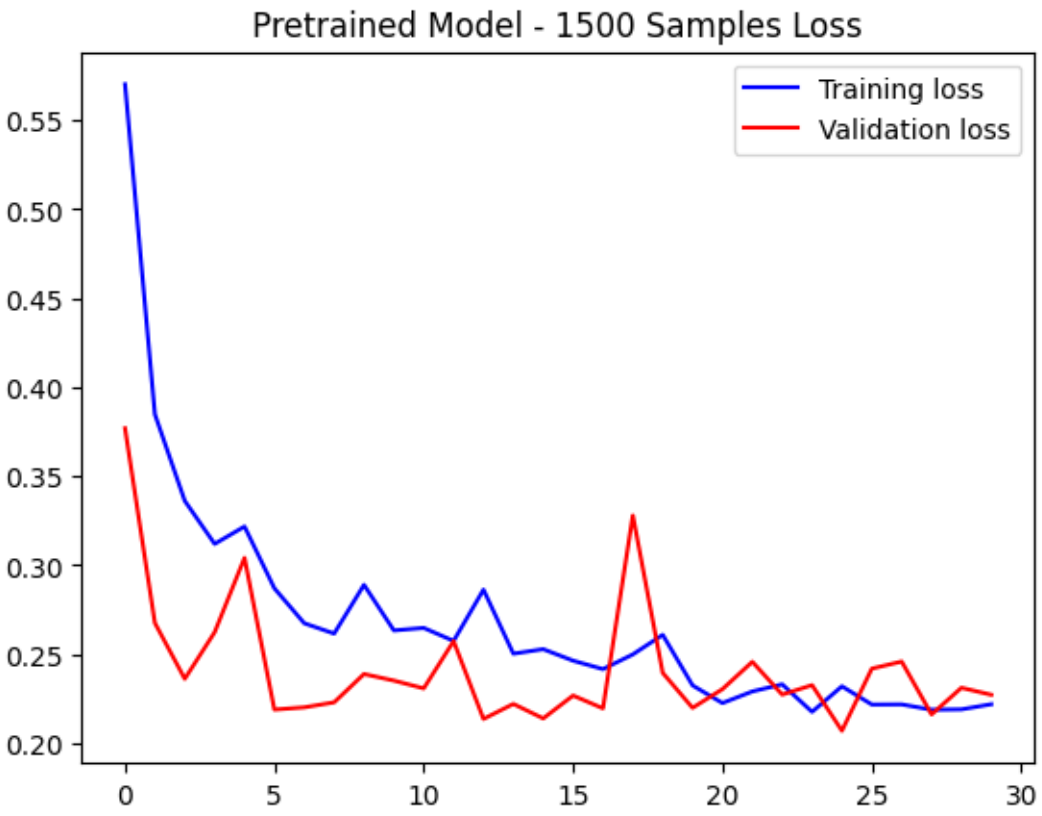

Model from Scratch - 2000 Samples Accuracy
Training acc
Validation acc
0.75
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30

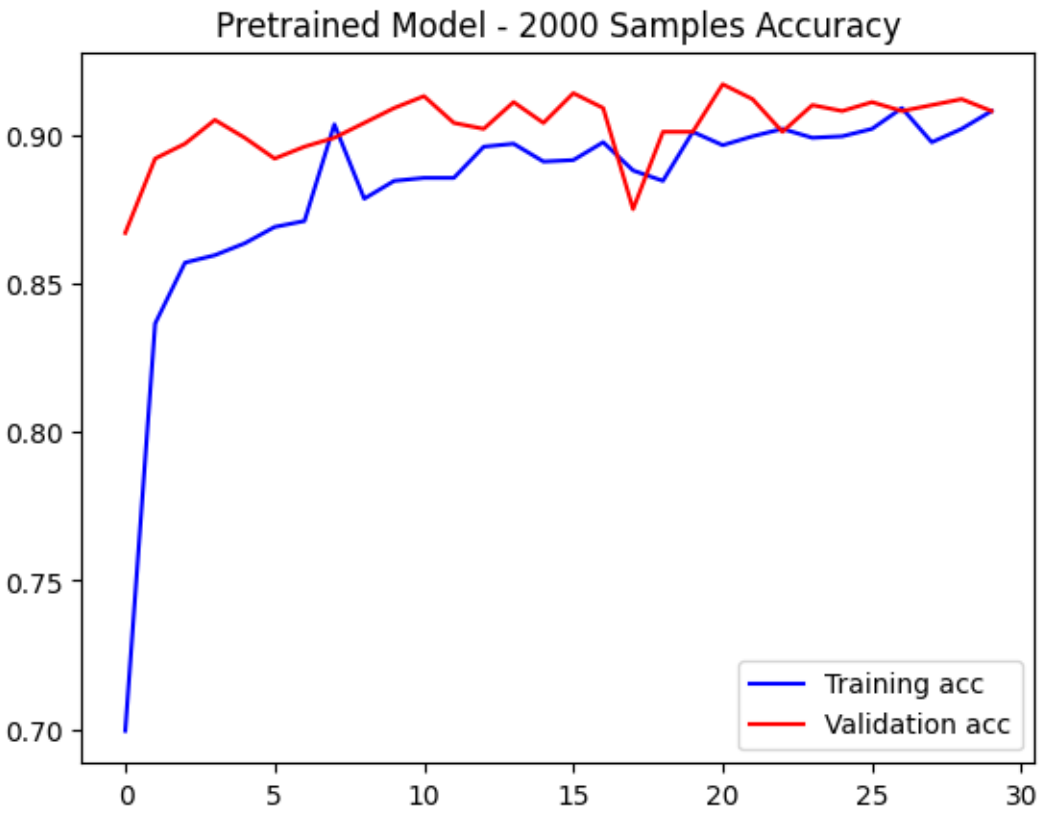

Model from Scratch - 2000 Samples Loss
Training loss
Validation loss
0.75
0.70
0.65
0.60
0.55
0
5
10
15
20
25
30

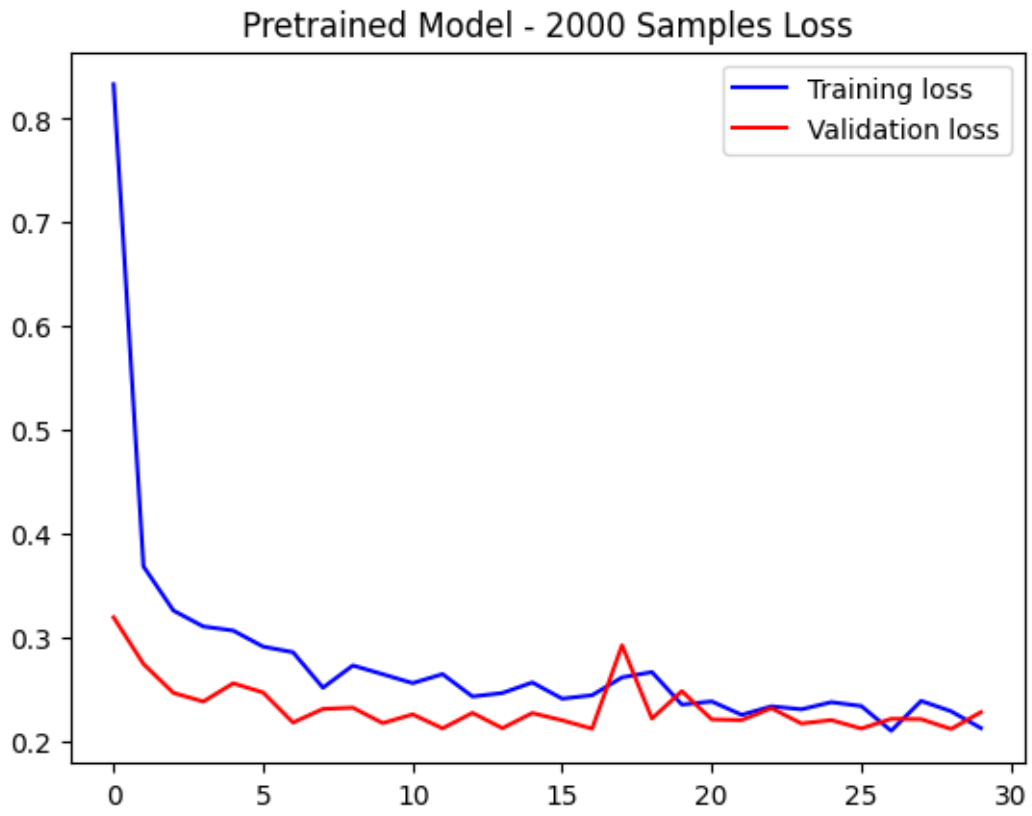

Pretrained Model - 1000 Samples Accuracy
0.925
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0
5
10
15
20
25
30
Training acc
Validation acc


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


Pretrained Model - 1500 Samples Accuracy
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0
5
10
15
20
25
30
Training acc
Validation acc


Pretrained Model - 1500 Samples Loss
Training loss
Validation loss
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0
5
10
15
20
25
30

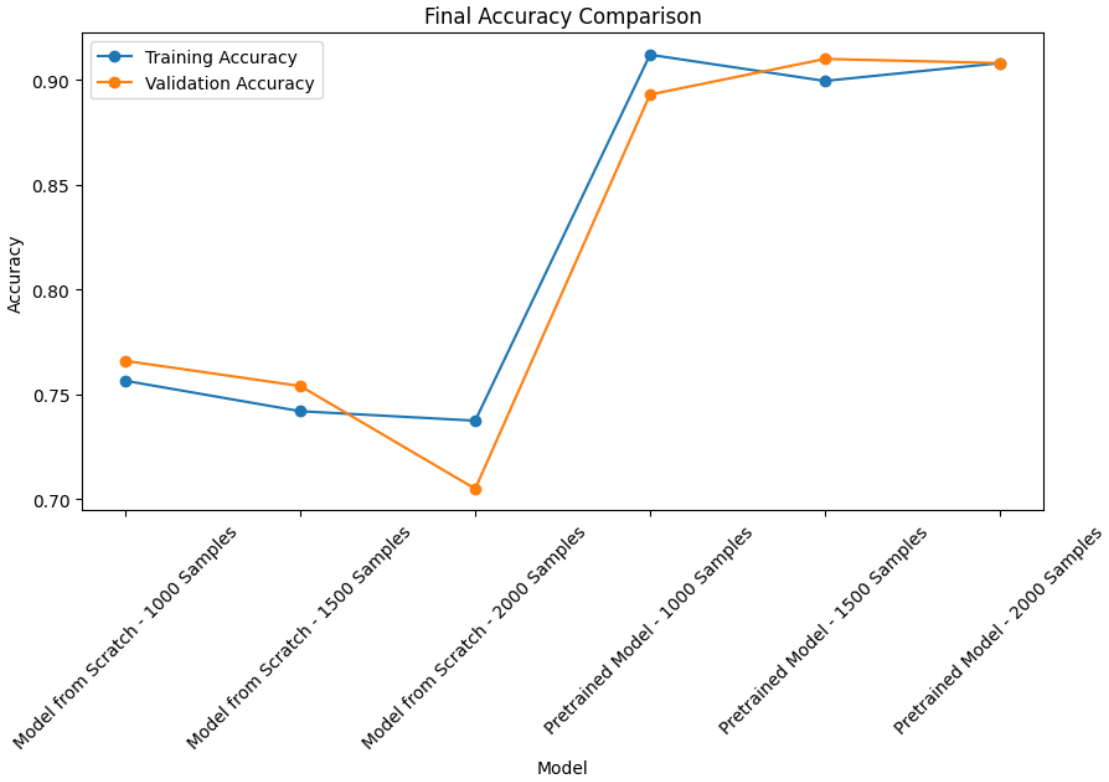

Pretrained Model - 2000 Samples Accuracy
0.90
0.85
0.80
0.75
0.70
0
5
10
15
20
25
30
Training acc
Validation acc

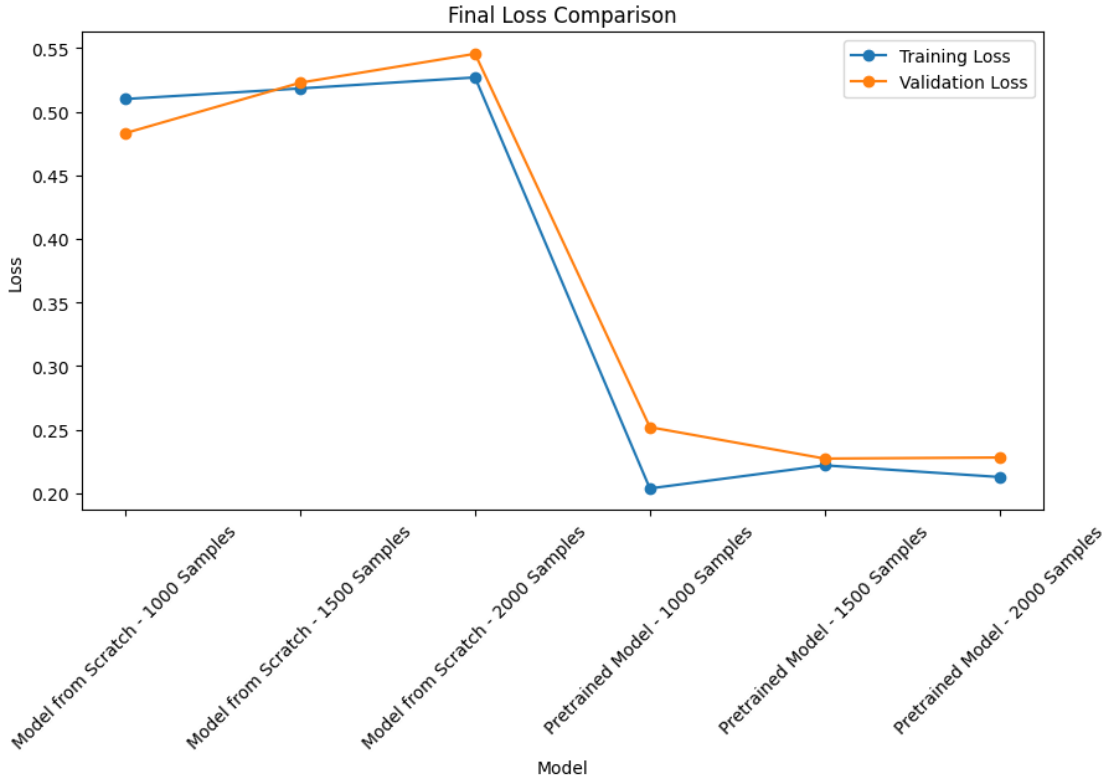


[33]:
```python
def aggregate_results(all_histories, model_labels):
    summary = {}

    for i, training_log in enumerate(all_histories):
        # Access history using the 'history' attribute directly
        final_train_acc = training_log.history['accuracy'][-1]
        final_val_acc = training_log.history['val_accuracy'][-1]
        final_train_loss = training_log.history['loss'][-1]
        final_val_loss = training_log.history['val_loss'][-1]

        summary[model_labels[i]] = {
            'Final Training Accuracy': final_train_acc,
            'Final Validation Accuracy': final_val_acc,
            'Final Training Loss': final_train_loss,
            'Final Validation Loss': final_val_loss,
        }

    return summary
```

[34]:
```python
# Function for comparing cnn_model's performances visually
def compare_models(final_scores):
```

```
    model_labels = list(final_scores.keys())
    train_acc = [final_scores[label]['Final Training Accuracy'] for label in␣
 ↪model_labels]
    val_acc = [final_scores[label]['Final Validation Accuracy'] for label in␣
 ↪model_labels]
    train_loss = [final_scores[label]['Final Training Loss'] for label in␣
 ↪model_labels]
    val_loss = [final_scores[label]['Final Validation Loss'] for label in␣
 ↪model_labels]

    # Plot for comparing Accuracy
    plt.figure(figsize=(10, 5))
    plt.plot(model_labels, train_acc, label='Training Accuracy', marker='o')
    plt.plot(model_labels, val_acc, label='Validation Accuracy', marker='o')
    plt.xticks(rotation=45)
    plt.title('Final Accuracy Comparison')
    plt.xlabel('Model')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()

    # Comparison plot for loss
    plt.figure(figsize=(10, 5))
    plt.plot(model_labels, train_loss, label='Training Loss', marker='o')
    plt.plot(model_labels, val_loss, label='Validation Loss', marker='o') #This␣
 ↪line was outside the function
    plt.xticks(rotation=45) # This line had incorrect indentation
    plt.title('Final Loss Comparison')
    plt.xlabel('Model')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

# Summary of final values
final_scores = aggregate_results(all_histories, model_labels) # Call␣
 ↪aggregate_results and assign the result to final_scores

# Plotting all the comparisons
compare_models(final_scores) # Now final_scores is defined
```


Final Accuracy Comparison
Training Accuracy
Validation Accuracy
Accuracy
0.90
0.85
0.80
0.75
0.70
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


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