covid19-detection-using-cnn

May 3, 2023

1 The task: Detect Covid-19 from Chest X-Ray Images using neural networks – (max ~ 1600words)

Subtask 1: Develop deep learning models to classify Covid-19 and Normal CXRs. - Convolutional neural networks (CNNs) will be used in this task. - For comparison purposes, purely feed forward neural networks (FNN) will also be applied to this problem. Compare the performance between CNN and FNN. Explain why CNNs would be more suitable for image classification (5 Marks). - You must explore generalization techniques, such as data augmentation, weight decay, early stopping,ensembles, and dropout. - You must demonstrate how each of these techniques was used, and justifywhich (maybe more than one of these techniques) would be the best generalization techniques for this task (10 Marks). - Report the performance in terms of accuracy of your models. - You must explain the network architecture used in your model, explain how you have monitored the convergence of the model, and how overfitting was prevented (10 Marks).

To accomplish this, use the 100 Covid-19 and 100 Normal CXRs images provided to you. Follow 80% (train) / 20% (test) process.

We must create two distinct models utilizing these architectures in order to compare the performance of CNN and FNN for categorizing Covid-19 and Normal CXRs.

Let's talk about why CNNs are more suited for image classification jobs than FNNs before continuing.

By applying filters to various areas of the image, CNNs are made to recognize features and patterns in images. The CNN can recognize regional patterns in the image, such as edges, corners, and textures, thanks to these filters, which it learns during the training phase.

FNNs, on the other hand, are made to function with data that has a grid-like structure, like tabular data or time series data. FNNs are not as good at spotting patterns in images since they don't include spatial relationships between pixels.

Therefore, CNNs are more suitable for image classification tasks because they can learn and detect complex features in images that are crucial for accurate classification.

1. Preprocessing the data : We need to preprocess the dataset to prepare it for training the CNN. This step involves resizing the images to a standard size, normalizing the pixel values, and splitting the dataset into training, validation, and testing sets.

[9]:
```
import os
import cv2
import numpy as np
```

```python
from sklearn.model_selection import train_test_split

# define the path to the dataset directory
DATASET_PATH = 'CS552J_DMDL_Assessment_1_Dataset'

# define the size of the images
IMG_SIZE = (224, 224)

# define the number of channels
CHANNELS = 3

# define the label categories
CATEGORIES = ['Covid-19', 'Normal']

# initialize the data and labels arrays
data = []
labels = []

# loop over the image paths
for category in CATEGORIES:
    path = os.path.join(DATASET_PATH, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = cv2.imread(img_path)
        image = cv2.resize(image, IMG_SIZE)
        image = np.array(image, dtype=np.float32)
        image /= 255.0
        data.append(image)
        labels.append(category)

# convert the data and labels to numpy arrays
data = np.array(data)
labels = np.array(labels)

# split the dataset into training, validation, and testing sets
(train_data, test_data, train_labels, test_labels) = train_test_split(data,␣
↪labels, test_size=0.2, random_state=42)
(train_data, val_data, train_labels, val_labels) = train_test_split(train_data,␣
↪train_labels, test_size=0.2, random_state=42)
```

[10]:
```python
print(train_labels)
```

```
['Covid-19' 'Covid-19' 'Covid-19' 'Normal' 'Covid-19' 'Covid-19' 'Normal'
 'Normal' 'Covid-19' 'Normal' 'Normal' 'Covid-19' 'Covid-19' 'Covid-19'
 'Normal' 'Covid-19' 'Normal' 'Covid-19' 'Covid-19' 'Normal' 'Normal'
 'Covid-19' 'Covid-19' 'Normal' 'Covid-19' 'Normal' 'Normal' 'Covid-19'
 'Normal' 'Normal' 'Normal' 'Covid-19' 'Covid-19' 'Normal' 'Normal'
```

```
'Normal' 'Covid-19' 'Covid-19' 'Covid-19' 'Normal' 'Normal' 'Normal'
'Normal' 'Covid-19' 'Covid-19' 'Covid-19' 'Normal' 'Covid-19' 'Normal'
'Covid-19' 'Normal' 'Normal' 'Covid-19' 'Normal' 'Normal' 'Covid-19'
'Covid-19' 'Covid-19' 'Normal' 'Covid-19' 'Normal' 'Covid-19' 'Covid-19'
'Normal' 'Normal' 'Covid-19' 'Normal' 'Covid-19' 'Normal' 'Covid-19'
'Covid-19' 'Covid-19' 'Normal' 'Covid-19' 'Covid-19' 'Normal' 'Normal'
'Covid-19' 'Normal' 'Normal' 'Normal' 'Normal' 'Covid-19' 'Normal'
'Normal' 'Normal' 'Covid-19' 'Normal' 'Covid-19' 'Normal' 'Normal'
'Covid-19' 'Covid-19' 'Covid-19' 'Normal' 'Normal' 'Covid-19' 'Covid-19'
'Normal' 'Covid-19' 'Normal' 'Covid-19' 'Covid-19' 'Covid-19' 'Normal'
'Covid-19' 'Covid-19' 'Covid-19' 'Covid-19' 'Covid-19' 'Normal' 'Normal'
'Normal' 'Normal' 'Covid-19' 'Covid-19' 'Normal' 'Normal' 'Normal'
'Covid-19' 'Normal' 'Normal' 'Normal' 'Covid-19' 'Covid-19' 'Normal'
'Covid-19' 'Normal']
```

[11]:
```
train_data.shape
```

[11]: (128, 224, 224, 3)

[12]:
```
test_data.shape
```

[12]: (40, 224, 224, 3)

[13]:
```
val_data.shape
```

[13]: (32, 224, 224, 3)

[14]:
```
# define label encoder
label_encoder = {'Covid-19': 1, 'Normal': 0}

# convert labels to numerical values
train_labels = np.array([label_encoder[label] for label in train_labels])
val_labels = np.array([label_encoder[label] for label in val_labels])
test_labels = np.array([label_encoder[label] for label in test_labels])
```

[15]:
```
# Limit GPU memory usage to avoid OOM errors
import tensorflow as tf

gpus = tf.config.experimental.list_physical_devices('GPU')
print(gpus)
if gpus:
    try:
        tf.config.experimental.set_virtual_device_configuration(
            gpus[0],
            [tf.config.experimental.
 ↪VirtualDeviceConfiguration(memory_limit=1024 * 6)]
        )
```

```
    except RuntimeError as e:
        print(e)

# Verify that the GPU is being used
print("Num GPUs Available: ", len(tf.config.experimental.
 ↪list_physical_devices('GPU')))
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
Num GPUs Available:  1
```

[16]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
 ↪Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras import regularizers

from tensorflow.keras.optimizers import Adam
```

2. Implementing 5 different models:

- model_cnn = CNN model with 50 epochs
- model_cnn1 = CNN model with regularization
- model_cnn2 = CNN model with learning rate and 100 epochs
- model_cnn3 = CNN model with early stopping
- model_cnn4 = CNN model with weight decay and data augmentation

[17]:
```
# CNN model0

model_cnn = Sequential()
model_cnn.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model_cnn.add(MaxPooling2D((2, 2)))
model_cnn.add(Conv2D(64, (3, 3), activation='relu'))
model_cnn.add(MaxPooling2D((2, 2)))
model_cnn.add(Conv2D(128, (3, 3), activation='relu'))
model_cnn.add(MaxPooling2D((2, 2)))
model_cnn.add(Conv2D(128, (3, 3), activation='relu'))
model_cnn.add(MaxPooling2D((2, 2)))
model_cnn.add(Flatten())
model_cnn.add(Dense(512, activation='relu'))
model_cnn.add(Dropout(0.5))
model_cnn.add(Dense(1, activation='sigmoid'))

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

history_cnn = model_cnn.fit(train_data, train_labels, batch_size=32, epochs=50,␣
 ↪validation_data=(val_data, val_labels))
```

```
WARNING:tensorflow:From c:\Users\abhij\.conda\envs\image_ML\lib\site-
packages\tensorflow\python\ops\nn_impl.py:180:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 128 samples, validate on 32 samples
Epoch 1/50
128/128 [==============================] - 2s 17ms/sample - loss: 0.6973 - acc:
0.6172 - val_loss: 0.6632 - val_acc: 0.5312
Epoch 2/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.6021 - acc:
0.6797 - val_loss: 0.4795 - val_acc: 0.9062
Epoch 3/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.3582 - acc:
0.8672 - val_loss: 0.3048 - val_acc: 0.8750
Epoch 4/50
128/128 [==============================] - 0s 3ms/sample - loss: 0.3912 - acc:
0.9062 - val_loss: 0.7490 - val_acc: 0.7500
Epoch 5/50
128/128 [==============================] - 0s 3ms/sample - loss: 0.2745 - acc:
0.8984 - val_loss: 0.3356 - val_acc: 0.8125
Epoch 6/50
128/128 [==============================] - 0s 3ms/sample - loss: 0.2571 - acc:
0.8906 - val_loss: 0.4727 - val_acc: 0.8125
Epoch 7/50
128/128 [==============================] - 0s 3ms/sample - loss: 0.2512 - acc:
0.8984 - val_loss: 0.1663 - val_acc: 0.9062
Epoch 8/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1416 - acc:
0.9375 - val_loss: 0.1227 - val_acc: 0.9375
Epoch 9/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1796 - acc:
0.9453 - val_loss: 0.2341 - val_acc: 0.9062
Epoch 10/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1259 - acc:
0.9609 - val_loss: 0.2564 - val_acc: 0.9062
Epoch 11/50
128/128 [==============================] - 0s 3ms/sample - loss: 0.1261 - acc:
0.9766 - val_loss: 0.0940 - val_acc: 0.9375
Epoch 12/50
128/128 [==============================] - 0s 3ms/sample - loss: 0.0872 - acc:
0.9766 - val_loss: 0.1821 - val_acc: 0.9375
Epoch 13/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0799 - acc:
0.9766 - val_loss: 0.1269 - val_acc: 0.9688
Epoch 14/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0466 - acc:
```

```
0.9766 - val_loss: 0.0783 - val_acc: 0.9688
Epoch 15/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0557 - acc:
0.9844 - val_loss: 0.0795 - val_acc: 0.9688
Epoch 16/50
128/128 [==============================] - 0s 3ms/sample - loss: 0.0410 - acc:
0.9844 - val_loss: 0.0889 - val_acc: 0.9688
Epoch 17/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0709 - acc:
0.9844 - val_loss: 0.0795 - val_acc: 0.9375
Epoch 18/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0294 - acc:
0.9922 - val_loss: 0.1202 - val_acc: 0.9375
Epoch 19/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0366 - acc:
0.9844 - val_loss: 0.1734 - val_acc: 0.9688
Epoch 20/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0214 - acc:
1.0000 - val_loss: 0.1361 - val_acc: 0.9688
Epoch 21/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0127 - acc:
1.0000 - val_loss: 0.1230 - val_acc: 0.9688
Epoch 22/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0487 - acc:
0.9844 - val_loss: 0.1004 - val_acc: 0.9375
Epoch 23/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0228 - acc:
0.9922 - val_loss: 0.1630 - val_acc: 0.9062
Epoch 24/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0450 - acc:
0.9922 - val_loss: 0.1113 - val_acc: 0.9375
Epoch 25/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0143 - acc:
0.9922 - val_loss: 0.2004 - val_acc: 0.9375
Epoch 26/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0107 - acc:
1.0000 - val_loss: 0.2058 - val_acc: 0.9375
Epoch 27/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0310 - acc:
0.9844 - val_loss: 0.2204 - val_acc: 0.9688
Epoch 28/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0109 - acc:
0.9922 - val_loss: 0.3035 - val_acc: 0.9688
Epoch 29/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0159 - acc:
0.9844 - val_loss: 0.4603 - val_acc: 0.9688
Epoch 30/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0061 - acc:
```

```
1.0000 - val_loss: 0.4248 - val_acc: 0.9062
Epoch 31/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0093 - acc:
1.0000 - val_loss: 0.4126 - val_acc: 0.9062
Epoch 32/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0093 - acc:
1.0000 - val_loss: 0.4263 - val_acc: 0.9062
Epoch 33/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0028 - acc:
1.0000 - val_loss: 0.4609 - val_acc: 0.9375
Epoch 34/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0011 - acc:
1.0000 - val_loss: 0.4923 - val_acc: 0.9688
Epoch 35/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0018 - acc:
1.0000 - val_loss: 0.4358 - val_acc: 0.9688
Epoch 36/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.0011 - acc:
1.0000 - val_loss: 0.3348 - val_acc: 0.9688
Epoch 37/50
128/128 [==============================] - 0s 2ms/sample - loss: 5.4602e-04 -
acc: 1.0000 - val_loss: 0.2643 - val_acc: 0.9688
Epoch 38/50
128/128 [==============================] - 0s 2ms/sample - loss: 9.4367e-04 -
acc: 1.0000 - val_loss: 0.2048 - val_acc: 0.9375
Epoch 39/50
128/128 [==============================] - 0s 2ms/sample - loss: 2.1356e-04 -
acc: 1.0000 - val_loss: 0.1659 - val_acc: 0.9375
Epoch 40/50
128/128 [==============================] - 0s 2ms/sample - loss: 1.8436e-04 -
acc: 1.0000 - val_loss: 0.1471 - val_acc: 0.9375
Epoch 41/50
128/128 [==============================] - 0s 2ms/sample - loss: 5.6879e-04 -
acc: 1.0000 - val_loss: 0.1419 - val_acc: 0.9062
Epoch 42/50
128/128 [==============================] - 0s 2ms/sample - loss: 5.9746e-04 -
acc: 1.0000 - val_loss: 0.1367 - val_acc: 0.9062
Epoch 43/50
128/128 [==============================] - 0s 2ms/sample - loss: 2.6983e-04 -
acc: 1.0000 - val_loss: 0.1305 - val_acc: 0.9375
Epoch 44/50
128/128 [==============================] - 0s 2ms/sample - loss: 1.7083e-04 -
acc: 1.0000 - val_loss: 0.1279 - val_acc: 0.9375
Epoch 45/50
128/128 [==============================] - 0s 2ms/sample - loss: 8.8644e-05 -
acc: 1.0000 - val_loss: 0.1298 - val_acc: 0.9375
Epoch 46/50
128/128 [==============================] - 0s 2ms/sample - loss: 2.1609e-04 -
```

```
acc: 1.0000 - val_loss: 0.1322 - val_acc: 0.9375
Epoch 47/50
128/128 [==============================] - 0s 2ms/sample - loss: 2.4483e-04 -
acc: 1.0000 - val_loss: 0.1342 - val_acc: 0.9375
Epoch 48/50
128/128 [==============================] - 0s 2ms/sample - loss: 9.2519e-05 -
acc: 1.0000 - val_loss: 0.1365 - val_acc: 0.9375
Epoch 49/50
128/128 [==============================] - 0s 2ms/sample - loss: 3.7104e-04 -
acc: 1.0000 - val_loss: 0.1319 - val_acc: 0.9375
Epoch 50/50
128/128 [==============================] - 0s 2ms/sample - loss: 3.7014e-04 -
acc: 1.0000 - val_loss: 0.1266 - val_acc: 0.9688
```

[18]:

```
# evaluate CNN model0
test_loss_cnn, test_acc_cnn = model_cnn.evaluate(test_data, test_labels)

print('Test accuracy:', test_acc_cnn)
print('Test loss:', test_loss_cnn)
```

```
40/40 [==============================] - 0s 3ms/sample - loss: 0.4657 - acc:
0.9250
Test accuracy: 0.925
Test loss: 0.46570894718170164
```

[19]:

```
# CNN model with regularization

model_cnn1 = Sequential()
model_cnn1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3),␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn1.add(MaxPooling2D((2, 2)))
model_cnn1.add(Conv2D(64, (3, 3), activation='relu',␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn1.add(MaxPooling2D((2, 2)))
model_cnn1.add(Conv2D(128, (3, 3), activation='relu',␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn1.add(MaxPooling2D((2, 2)))
model_cnn1.add(Conv2D(128, (3, 3), activation='relu',␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn1.add(MaxPooling2D((2, 2)))
model_cnn1.add(Flatten())
model_cnn1.add(Dense(512, activation='relu', kernel_regularizer=regularizers.
 ↪l2(0.001)))
model_cnn1.add(Dropout(0.5))
model_cnn1.add(Dense(1, activation='sigmoid'))
```

```python
model_cnn1.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])

history_cnn1 = model_cnn1.fit(train_data, train_labels, batch_size=32,␣
↪epochs=50, validation_data=(val_data, val_labels))
```

```
Train on 128 samples, validate on 32 samples
Epoch 1/50
128/128 [==============================] - 1s 12ms/sample - loss: 2.0259 - acc:
0.4688 - val_loss: 1.5887 - val_acc: 0.5312
Epoch 2/50
128/128 [==============================] - 0s 2ms/sample - loss: 1.5007 - acc:
0.5234 - val_loss: 1.3389 - val_acc: 0.5312
Epoch 3/50
128/128 [==============================] - 0s 2ms/sample - loss: 1.2695 - acc:
0.5078 - val_loss: 1.1615 - val_acc: 0.9688
Epoch 4/50
128/128 [==============================] - 0s 2ms/sample - loss: 1.0636 - acc:
0.8984 - val_loss: 0.9433 - val_acc: 0.9375
Epoch 5/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.7772 - acc:
0.9141 - val_loss: 0.7099 - val_acc: 0.8438
Epoch 6/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.6087 - acc:
0.9297 - val_loss: 0.7016 - val_acc: 0.8438
Epoch 7/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.6618 - acc:
0.9141 - val_loss: 0.8283 - val_acc: 0.8125
Epoch 8/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.5123 - acc:
0.9453 - val_loss: 0.6239 - val_acc: 0.8125
Epoch 9/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.4816 - acc:
0.9609 - val_loss: 0.8893 - val_acc: 0.8125
Epoch 10/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.4685 - acc:
0.9609 - val_loss: 0.5784 - val_acc: 0.9062
Epoch 11/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.4546 - acc:
0.9531 - val_loss: 0.6162 - val_acc: 0.8438
Epoch 12/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.4073 - acc:
0.9766 - val_loss: 0.6054 - val_acc: 0.8750
Epoch 13/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.3737 - acc:
0.9766 - val_loss: 0.5328 - val_acc: 0.9062
Epoch 14/50
```

```
128/128 [==============================] - 0s 2ms/sample - loss: 0.3739 - acc:
0.9688 - val_loss: 0.6196 - val_acc: 0.8750
Epoch 15/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.3488 - acc:
0.9688 - val_loss: 0.5219 - val_acc: 0.8750
Epoch 16/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.3331 - acc:
0.9688 - val_loss: 0.4584 - val_acc: 0.8750
Epoch 17/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.3216 - acc:
0.9766 - val_loss: 0.4895 - val_acc: 0.9062
Epoch 18/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.3171 - acc:
0.9766 - val_loss: 0.4548 - val_acc: 0.8438
Epoch 19/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2987 - acc:
0.9688 - val_loss: 0.4243 - val_acc: 0.9062
Epoch 20/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.3016 - acc:
0.9766 - val_loss: 0.3868 - val_acc: 0.9062
Epoch 21/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2614 - acc:
0.9844 - val_loss: 0.3753 - val_acc: 0.9375
Epoch 22/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2626 - acc:
0.9922 - val_loss: 0.5384 - val_acc: 0.8438
Epoch 23/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2690 - acc:
0.9688 - val_loss: 0.3771 - val_acc: 0.9062
Epoch 24/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2428 - acc:
0.9844 - val_loss: 0.3251 - val_acc: 0.9062
Epoch 25/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2403 - acc:
0.9844 - val_loss: 0.4051 - val_acc: 0.9062
Epoch 26/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2212 - acc:
0.9922 - val_loss: 0.5235 - val_acc: 0.8750
Epoch 27/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.3089 - acc:
0.9766 - val_loss: 0.5366 - val_acc: 0.8750
Epoch 28/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2805 - acc:
0.9531 - val_loss: 0.3691 - val_acc: 0.9375
Epoch 29/50
128/128 [==============================] - 0s 3ms/sample - loss: 0.3002 - acc:
0.9531 - val_loss: 0.5040 - val_acc: 0.8438
Epoch 30/50
```

```
128/128 [==============================] - 0s 2ms/sample - loss: 0.2922 - acc:
0.9766 - val_loss: 0.4135 - val_acc: 0.8750
Epoch 31/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2336 - acc:
0.9844 - val_loss: 0.3087 - val_acc: 0.9062
Epoch 32/50
128/128 [==============================] - 0s 3ms/sample - loss: 0.2815 - acc:
0.9844 - val_loss: 0.3651 - val_acc: 0.9375
Epoch 33/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2409 - acc:
0.9766 - val_loss: 0.4559 - val_acc: 0.8750
Epoch 34/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2160 - acc:
1.0000 - val_loss: 0.5039 - val_acc: 0.8750
Epoch 35/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.2146 - acc:
0.9922 - val_loss: 0.4400 - val_acc: 0.9062
Epoch 36/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1991 - acc:
1.0000 - val_loss: 0.4298 - val_acc: 0.8750
Epoch 37/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1996 - acc:
0.9922 - val_loss: 0.4304 - val_acc: 0.8750
Epoch 38/50
128/128 [==============================] - 0s 3ms/sample - loss: 0.1819 - acc:
1.0000 - val_loss: 0.3602 - val_acc: 0.9688
Epoch 39/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1755 - acc:
1.0000 - val_loss: 0.3630 - val_acc: 0.9688
Epoch 40/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1861 - acc:
0.9844 - val_loss: 0.3637 - val_acc: 0.8750
Epoch 41/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1679 - acc:
1.0000 - val_loss: 0.4035 - val_acc: 0.8750
Epoch 42/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1748 - acc:
0.9922 - val_loss: 0.4416 - val_acc: 0.8750
Epoch 43/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1680 - acc:
1.0000 - val_loss: 0.4835 - val_acc: 0.9062
Epoch 44/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1649 - acc:
0.9922 - val_loss: 0.3233 - val_acc: 0.9062
Epoch 45/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1606 - acc:
0.9922 - val_loss: 0.3104 - val_acc: 0.8750
Epoch 46/50
```

```
128/128 [==============================] - 0s 2ms/sample - loss: 0.1587 - acc:
1.0000 - val_loss: 0.2922 - val_acc: 0.9688
Epoch 47/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1495 - acc:
1.0000 - val_loss: 0.3009 - val_acc: 0.9688
Epoch 48/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1460 - acc:
1.0000 - val_loss: 0.3018 - val_acc: 0.9688
Epoch 49/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1522 - acc:
0.9922 - val_loss: 0.2787 - val_acc: 0.9688
Epoch 50/50
128/128 [==============================] - 0s 2ms/sample - loss: 0.1418 - acc:
1.0000 - val_loss: 0.2294 - val_acc: 0.9062
```

[20]:

```
# evaluate CNN model1
test_loss_cnn, test_acc_cnn = model_cnn1.evaluate(test_data, test_labels)

print('Test accuracy:', test_acc_cnn)
print('Test loss:', test_loss_cnn)
```

```
40/40 [==============================] - 0s 1ms/sample - loss: 0.2614 - acc:
0.9250
Test accuracy: 0.925
Test loss: 0.261447811126709
```

[21]:

```
# CNN model with Learning Rate = 0.0001 and epochs = 100

model_cnn2 = Sequential()
model_cnn2.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3),␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn2.add(MaxPooling2D((2, 2)))
model_cnn2.add(Conv2D(64, (3, 3), activation='relu',␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn2.add(MaxPooling2D((2, 2)))
model_cnn2.add(Conv2D(128, (3, 3), activation='relu',␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn2.add(MaxPooling2D((2, 2)))
model_cnn2.add(Conv2D(128, (3, 3), activation='relu',␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn2.add(MaxPooling2D((2, 2)))
model_cnn2.add(Flatten())
model_cnn2.add(Dense(512, activation='relu', kernel_regularizer=regularizers.
 ↪l2(0.001)))
model_cnn2.add(Dropout(0.5))
model_cnn2.add(Dense(1, activation='sigmoid'))
```

```
opt = Adam(lr=0.0001)

model_cnn2.compile(optimizer=opt, loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])

history_cnn2 = model_cnn2.fit(train_data, train_labels, batch_size=32,␣
 ↪epochs=100, validation_data=(val_data, val_labels))
```

```
Train on 128 samples, validate on 32 samples
Epoch 1/100
128/128 [==============================] - 2s 13ms/sample - loss: 1.9381 - acc:
0.4453 - val_loss: 1.8776 - val_acc: 0.5312
Epoch 2/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.8498 - acc:
0.8281 - val_loss: 1.8124 - val_acc: 0.9375
Epoch 3/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.7789 - acc:
0.7891 - val_loss: 1.7223 - val_acc: 0.6875
Epoch 4/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.6880 - acc:
0.8438 - val_loss: 1.6318 - val_acc: 0.9062
Epoch 5/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.5742 - acc:
0.9062 - val_loss: 1.5201 - val_acc: 0.9062
Epoch 6/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.4648 - acc:
0.9141 - val_loss: 1.3978 - val_acc: 0.9688
Epoch 7/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.3159 - acc:
0.9766 - val_loss: 1.2789 - val_acc: 0.9062
Epoch 8/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.2141 - acc:
0.9219 - val_loss: 1.2293 - val_acc: 0.8750
Epoch 9/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.1362 - acc:
0.9609 - val_loss: 1.1002 - val_acc: 0.9062
Epoch 10/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.0714 - acc:
0.9453 - val_loss: 1.0458 - val_acc: 0.9688
Epoch 11/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.0199 - acc:
0.9453 - val_loss: 1.0139 - val_acc: 0.9375
Epoch 12/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.0156 - acc:
0.9219 - val_loss: 0.9745 - val_acc: 0.9688
Epoch 13/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.9790 - acc:
```

```
0.9609 - val_loss: 0.9431 - val_acc: 0.9062
Epoch 14/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.9204 - acc:
0.9531 - val_loss: 0.9910 - val_acc: 0.9688
Epoch 15/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.9213 - acc:
0.9609 - val_loss: 0.9024 - val_acc: 0.9062
Epoch 16/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.9063 - acc:
0.9531 - val_loss: 0.8837 - val_acc: 0.9375
Epoch 17/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.8819 - acc:
0.9531 - val_loss: 0.9134 - val_acc: 0.9688
Epoch 18/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.8523 - acc:
0.9609 - val_loss: 0.8945 - val_acc: 0.9375
Epoch 19/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.8438 - acc:
0.9688 - val_loss: 0.8460 - val_acc: 0.9375
Epoch 20/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.8006 - acc:
0.9609 - val_loss: 0.8851 - val_acc: 0.9688
Epoch 21/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7892 - acc:
0.9609 - val_loss: 0.8352 - val_acc: 0.9062
Epoch 22/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7658 - acc:
0.9688 - val_loss: 0.8066 - val_acc: 0.9062
Epoch 23/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7474 - acc:
0.9688 - val_loss: 0.8147 - val_acc: 0.9375
Epoch 24/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7229 - acc:
0.9688 - val_loss: 0.7795 - val_acc: 0.9375
Epoch 25/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.7195 - acc:
0.9766 - val_loss: 0.7699 - val_acc: 0.9062
Epoch 26/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.6968 - acc:
0.9609 - val_loss: 0.7621 - val_acc: 0.9375
Epoch 27/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.6739 - acc:
0.9766 - val_loss: 0.7390 - val_acc: 0.9375
Epoch 28/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.6704 - acc:
0.9688 - val_loss: 0.7457 - val_acc: 0.9375
Epoch 29/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.6697 - acc:
```

```
0.9688 - val_loss: 0.7153 - val_acc: 0.9375
Epoch 30/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.6859 - acc:
0.9531 - val_loss: 0.7112 - val_acc: 0.9375
Epoch 31/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.6310 - acc:
0.9688 - val_loss: 0.6961 - val_acc: 0.9062
Epoch 32/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.6114 - acc:
0.9844 - val_loss: 0.6849 - val_acc: 0.9375
Epoch 33/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.6292 - acc:
0.9766 - val_loss: 0.6844 - val_acc: 0.9375
Epoch 34/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5868 - acc:
0.9844 - val_loss: 0.6638 - val_acc: 0.9375
Epoch 35/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5931 - acc:
0.9766 - val_loss: 0.6578 - val_acc: 0.9375
Epoch 36/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5901 - acc:
0.9688 - val_loss: 0.6548 - val_acc: 0.9375
Epoch 37/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5886 - acc:
0.9844 - val_loss: 0.6645 - val_acc: 0.9688
Epoch 38/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5706 - acc:
0.9688 - val_loss: 0.6258 - val_acc: 0.9375
Epoch 39/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5481 - acc:
0.9844 - val_loss: 0.6183 - val_acc: 0.9375
Epoch 40/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5701 - acc:
0.9766 - val_loss: 0.6098 - val_acc: 0.9375
Epoch 41/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5507 - acc:
0.9766 - val_loss: 0.6064 - val_acc: 0.9375
Epoch 42/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5436 - acc:
0.9844 - val_loss: 0.6006 - val_acc: 0.9375
Epoch 43/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5775 - acc:
0.9688 - val_loss: 0.6037 - val_acc: 0.9375
Epoch 44/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5257 - acc:
0.9844 - val_loss: 0.6042 - val_acc: 0.9375
Epoch 45/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5374 - acc:
```

```
0.9844 - val_loss: 0.5800 - val_acc: 0.9688
Epoch 46/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5192 - acc:
0.9766 - val_loss: 0.5831 - val_acc: 0.9375
Epoch 47/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5045 - acc:
0.9922 - val_loss: 0.5834 - val_acc: 0.9688
Epoch 48/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4976 - acc:
0.9844 - val_loss: 0.5810 - val_acc: 0.9375
Epoch 49/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5084 - acc:
0.9844 - val_loss: 0.5610 - val_acc: 0.9688
Epoch 50/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4977 - acc:
0.9766 - val_loss: 0.5904 - val_acc: 0.9688
Epoch 51/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4866 - acc:
0.9844 - val_loss: 0.5673 - val_acc: 0.9688
Epoch 52/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4733 - acc:
0.9922 - val_loss: 0.5616 - val_acc: 0.9688
Epoch 53/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5039 - acc:
0.9688 - val_loss: 0.5768 - val_acc: 0.9375
Epoch 54/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4743 - acc:
0.9922 - val_loss: 0.5706 - val_acc: 0.9375
Epoch 55/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4807 - acc:
0.9844 - val_loss: 0.5497 - val_acc: 0.9688
Epoch 56/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4701 - acc:
0.9844 - val_loss: 0.5456 - val_acc: 0.9375
Epoch 57/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4561 - acc:
0.9922 - val_loss: 0.5525 - val_acc: 0.9688
Epoch 58/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4612 - acc:
0.9766 - val_loss: 0.5473 - val_acc: 0.9688
Epoch 59/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4426 - acc:
0.9922 - val_loss: 0.5231 - val_acc: 0.9688
Epoch 60/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4449 - acc:
0.9922 - val_loss: 0.5260 - val_acc: 0.9688
Epoch 61/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4571 - acc:
```

```
0.9844 - val_loss: 0.5126 - val_acc: 0.9688
Epoch 62/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4456 - acc:
0.9844 - val_loss: 0.5088 - val_acc: 0.9688
Epoch 63/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4365 - acc:
0.9844 - val_loss: 0.5574 - val_acc: 0.9375
Epoch 64/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4404 - acc:
0.9844 - val_loss: 0.5032 - val_acc: 0.9688
Epoch 65/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4254 - acc:
1.0000 - val_loss: 0.5015 - val_acc: 0.9688
Epoch 66/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4206 - acc:
0.9922 - val_loss: 0.5054 - val_acc: 0.9688
Epoch 67/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4149 - acc:
1.0000 - val_loss: 0.5019 - val_acc: 0.9688
Epoch 68/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4075 - acc:
1.0000 - val_loss: 0.5129 - val_acc: 0.9688
Epoch 69/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4163 - acc:
0.9922 - val_loss: 0.5279 - val_acc: 0.9688
Epoch 70/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4082 - acc:
0.9922 - val_loss: 0.4895 - val_acc: 0.9688
Epoch 71/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4141 - acc:
0.9922 - val_loss: 0.4859 - val_acc: 0.9688
Epoch 72/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3990 - acc:
1.0000 - val_loss: 0.5769 - val_acc: 0.9375
Epoch 73/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4056 - acc:
0.9922 - val_loss: 0.4767 - val_acc: 0.9688
Epoch 74/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3934 - acc:
0.9922 - val_loss: 0.4529 - val_acc: 0.9688
Epoch 75/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4033 - acc:
0.9922 - val_loss: 0.4615 - val_acc: 0.9688
Epoch 76/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3957 - acc:
0.9922 - val_loss: 0.4989 - val_acc: 0.9688
Epoch 77/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3873 - acc:
```

```
0.9922 - val_loss: 0.4643 - val_acc: 0.9688
Epoch 78/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3807 - acc:
1.0000 - val_loss: 0.4592 - val_acc: 0.9688
Epoch 79/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3892 - acc:
1.0000 - val_loss: 0.4801 - val_acc: 0.9688
Epoch 80/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3927 - acc:
0.9922 - val_loss: 0.5068 - val_acc: 0.9688
Epoch 81/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3750 - acc:
1.0000 - val_loss: 0.4477 - val_acc: 0.9688
Epoch 82/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3788 - acc:
0.9922 - val_loss: 0.4392 - val_acc: 0.9688
Epoch 83/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3690 - acc:
1.0000 - val_loss: 0.4969 - val_acc: 0.9688
Epoch 84/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3743 - acc:
0.9922 - val_loss: 0.5059 - val_acc: 0.9688
Epoch 85/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3641 - acc:
1.0000 - val_loss: 0.4522 - val_acc: 0.9688
Epoch 86/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3660 - acc:
1.0000 - val_loss: 0.4480 - val_acc: 0.9688
Epoch 87/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3593 - acc:
1.0000 - val_loss: 0.4865 - val_acc: 0.9688
Epoch 88/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3673 - acc:
0.9922 - val_loss: 0.4359 - val_acc: 0.9688
Epoch 89/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3599 - acc:
1.0000 - val_loss: 0.4224 - val_acc: 0.9688
Epoch 90/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3534 - acc:
1.0000 - val_loss: 0.4293 - val_acc: 0.9688
Epoch 91/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3587 - acc:
0.9922 - val_loss: 0.4135 - val_acc: 0.9688
Epoch 92/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3520 - acc:
1.0000 - val_loss: 0.4313 - val_acc: 0.9688
Epoch 93/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3544 - acc:
```

```
1.0000 - val_loss: 0.4239 - val_acc: 0.9688
Epoch 94/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3449 - acc:
1.0000 - val_loss: 0.3963 - val_acc: 0.9688
Epoch 95/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3485 - acc:
0.9922 - val_loss: 0.4509 - val_acc: 0.9688
Epoch 96/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3417 - acc:
1.0000 - val_loss: 0.4626 - val_acc: 0.9688
Epoch 97/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3441 - acc:
1.0000 - val_loss: 0.4078 - val_acc: 0.9688
Epoch 98/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3450 - acc:
0.9922 - val_loss: 0.4297 - val_acc: 0.9688
Epoch 99/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3348 - acc:
1.0000 - val_loss: 0.5020 - val_acc: 0.9375
Epoch 100/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3359 - acc:
1.0000 - val_loss: 0.4518 - val_acc: 0.9688
```

[22]:
```
# evaluate CNN model2
test_loss_cnn, test_acc_cnn = model_cnn2.evaluate(test_data, test_labels)

print('Test accuracy:', test_acc_cnn)
print('Test loss:', test_loss_cnn)
```

```
40/40 [==============================] - 0s 1ms/sample - loss: 0.3830 - acc:
1.0000
Test accuracy: 1.0
Test loss: 0.38304530978202 82
```

[23]:
```
# CNN model with lr and early stopping

model_cnn3 = Sequential()
model_cnn3.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3),␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn3.add(MaxPooling2D((2, 2)))
model_cnn3.add(Conv2D(64, (3, 3), activation='relu',␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn3.add(MaxPooling2D((2, 2)))
model_cnn3.add(Conv2D(128, (3, 3), activation='relu',␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn3.add(MaxPooling2D((2, 2)))
```

```
model_cnn3.add(Conv2D(128, (3, 3), activation='relu',␣
 ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn3.add(MaxPooling2D((2, 2)))
model_cnn3.add(Flatten())
model_cnn3.add(Dense(512, activation='relu', kernel_regularizer=regularizers.
 ↪l2(0.001)))
model_cnn3.add(Dropout(0.5))
model_cnn3.add(Dense(1, activation='sigmoid'))

opt = Adam(lr=0.0001)

model_cnn3.compile(optimizer=opt, loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])

es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)

history_cnn3 = model_cnn3.fit(train_data, train_labels, batch_size=32,␣
 ↪epochs=100, validation_data=(val_data, val_labels), callbacks=[es])
```

```
Train on 128 samples, validate on 32 samples
Epoch 1/100
128/128 [==============================] - 2s 12ms/sample - loss: 1.9251 - acc:
0.4688 - val_loss: 1.8635 - val_acc: 0.6250
Epoch 2/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.8313 - acc:
0.7266 - val_loss: 1.7840 - val_acc: 0.9062
Epoch 3/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.7273 - acc:
0.8750 - val_loss: 1.6705 - val_acc: 0.7188
Epoch 4/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.6566 - acc:
0.7344 - val_loss: 1.5542 - val_acc: 0.9688
Epoch 5/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.5110 - acc:
0.8984 - val_loss: 1.4444 - val_acc: 0.9375
Epoch 6/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.3962 - acc:
0.9297 - val_loss: 1.3160 - val_acc: 0.9688
Epoch 7/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.2858 - acc:
0.9375 - val_loss: 1.2188 - val_acc: 0.9688
Epoch 8/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.2104 - acc:
0.9297 - val_loss: 1.1509 - val_acc: 0.9688
Epoch 9/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.1552 - acc:
0.9141 - val_loss: 1.0897 - val_acc: 0.9375
```

```
Epoch 10/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.1584 - acc:
0.9297 - val_loss: 1.0669 - val_acc: 0.9375
Epoch 11/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.1306 - acc:
0.9141 - val_loss: 1.1199 - val_acc: 0.9062
Epoch 12/100
128/128 [==============================] - 0s 2ms/sample - loss: 1.0435 - acc:
0.9453 - val_loss: 1.0596 - val_acc: 0.9375
Epoch 13/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.9804 - acc:
0.9688 - val_loss: 0.9926 - val_acc: 0.9375
Epoch 14/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.9452 - acc:
0.9453 - val_loss: 0.9787 - val_acc: 0.9375
Epoch 15/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.9201 - acc:
0.9688 - val_loss: 0.9407 - val_acc: 0.9375
Epoch 16/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.9070 - acc:
0.9688 - val_loss: 0.9183 - val_acc: 0.9375
Epoch 17/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.8711 - acc:
0.9531 - val_loss: 0.8953 - val_acc: 0.9375
Epoch 18/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.8540 - acc:
0.9688 - val_loss: 0.8755 - val_acc: 0.9375
Epoch 19/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.8187 - acc:
0.9609 - val_loss: 0.8638 - val_acc: 0.9375
Epoch 20/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7993 - acc:
0.9688 - val_loss: 0.8598 - val_acc: 0.9375
Epoch 21/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7803 - acc:
0.9609 - val_loss: 0.8264 - val_acc: 0.9062
Epoch 22/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7744 - acc:
0.9688 - val_loss: 0.8438 - val_acc: 0.9375
Epoch 23/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7512 - acc:
0.9766 - val_loss: 0.8039 - val_acc: 0.9375
Epoch 24/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7420 - acc:
0.9766 - val_loss: 0.8008 - val_acc: 0.9375
Epoch 25/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7278 - acc:
0.9609 - val_loss: 0.7823 - val_acc: 0.9062
```

```
Epoch 26/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7208 - acc:
0.9688 - val_loss: 0.7609 - val_acc: 0.9375
Epoch 27/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.6772 - acc:
0.9766 - val_loss: 0.7815 - val_acc: 0.9375
Epoch 28/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7046 - acc:
0.9766 - val_loss: 0.7423 - val_acc: 0.9375
Epoch 29/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.6831 - acc:
0.9688 - val_loss: 0.7445 - val_acc: 0.9375
Epoch 30/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.7096 - acc:
0.9609 - val_loss: 0.7211 - val_acc: 0.9375
Epoch 31/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.6764 - acc:
0.9531 - val_loss: 0.7190 - val_acc: 0.9375
Epoch 32/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.6483 - acc:
0.9688 - val_loss: 0.7076 - val_acc: 0.9375
Epoch 33/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.6096 - acc:
0.9922 - val_loss: 0.7041 - val_acc: 0.9375
Epoch 34/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.6168 - acc:
0.9844 - val_loss: 0.6942 - val_acc: 0.9375
Epoch 35/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.6112 - acc:
0.9688 - val_loss: 0.6839 - val_acc: 0.9375
Epoch 36/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5915 - acc:
0.9844 - val_loss: 0.6705 - val_acc: 0.9375
Epoch 37/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5949 - acc:
0.9688 - val_loss: 0.6732 - val_acc: 0.9375
Epoch 38/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.6118 - acc:
0.9688 - val_loss: 0.7017 - val_acc: 0.9062
Epoch 39/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.6123 - acc:
0.9609 - val_loss: 0.6828 - val_acc: 0.9375
Epoch 40/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5945 - acc:
0.9766 - val_loss: 0.6617 - val_acc: 0.9375
Epoch 41/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5796 - acc:
0.9688 - val_loss: 0.7069 - val_acc: 0.9062
```

```
Epoch 42/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5620 - acc:
0.9844 - val_loss: 0.6765 - val_acc: 0.9375
Epoch 43/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.6058 - acc:
0.9688 - val_loss: 0.7132 - val_acc: 0.9375
Epoch 44/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5491 - acc:
0.9844 - val_loss: 0.7203 - val_acc: 0.9062
Epoch 45/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5948 - acc:
0.9531 - val_loss: 0.6268 - val_acc: 0.9375
Epoch 46/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5710 - acc:
0.9688 - val_loss: 0.6187 - val_acc: 0.9375
Epoch 47/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5201 - acc:
0.9922 - val_loss: 0.6130 - val_acc: 0.9375
Epoch 48/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5363 - acc:
0.9766 - val_loss: 0.6077 - val_acc: 0.9375
Epoch 49/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5405 - acc:
0.9766 - val_loss: 0.6062 - val_acc: 0.9375
Epoch 50/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5246 - acc:
0.9688 - val_loss: 0.5946 - val_acc: 0.9375
Epoch 51/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5052 - acc:
0.9844 - val_loss: 0.5961 - val_acc: 0.9375
Epoch 52/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5146 - acc:
0.9766 - val_loss: 0.5879 - val_acc: 0.9375
Epoch 53/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5414 - acc:
0.9766 - val_loss: 0.5709 - val_acc: 0.9375
Epoch 54/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.5121 - acc:
0.9766 - val_loss: 0.5789 - val_acc: 0.9375
Epoch 55/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.5243 - acc:
0.9688 - val_loss: 0.5752 - val_acc: 0.9375
Epoch 56/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4922 - acc:
0.9688 - val_loss: 0.5974 - val_acc: 0.9375
Epoch 57/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.4858 - acc:
0.9922 - val_loss: 0.5916 - val_acc: 0.9375
```

```
Epoch 58/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4845 - acc:
0.9922 - val_loss: 0.5955 - val_acc: 0.9062
Epoch 59/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.4852 - acc:
0.9844 - val_loss: 0.5699 - val_acc: 0.9688
Epoch 60/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.4668 - acc:
0.9922 - val_loss: 0.5865 - val_acc: 0.9375
Epoch 61/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.4687 - acc:
0.9844 - val_loss: 0.5789 - val_acc: 0.9375
Epoch 62/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.4636 - acc:
0.9922 - val_loss: 0.5617 - val_acc: 0.9688
Epoch 63/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.4665 - acc:
0.9766 - val_loss: 0.5659 - val_acc: 0.9688
Epoch 64/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.4465 - acc:
1.0000 - val_loss: 0.5694 - val_acc: 0.9688
Epoch 65/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4400 - acc:
1.0000 - val_loss: 0.5569 - val_acc: 0.9688
Epoch 66/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.4442 - acc:
0.9922 - val_loss: 0.5456 - val_acc: 0.9688
Epoch 67/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4351 - acc:
1.0000 - val_loss: 0.5423 - val_acc: 0.9688
Epoch 68/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4449 - acc:
0.9844 - val_loss: 0.5421 - val_acc: 0.9688
Epoch 69/100
128/128 [==============================] - 0s 3ms/sample - loss: 0.4336 - acc:
1.0000 - val_loss: 0.5325 - val_acc: 0.9688
Epoch 70/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4411 - acc:
0.9922 - val_loss: 0.5715 - val_acc: 0.9375
Epoch 71/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4415 - acc:
0.9844 - val_loss: 0.5315 - val_acc: 0.9688
Epoch 72/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4175 - acc:
1.0000 - val_loss: 0.5125 - val_acc: 0.9688
Epoch 73/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4423 - acc:
0.9766 - val_loss: 0.4992 - val_acc: 0.9688
```

```
Epoch 74/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4450 - acc:
0.9922 - val_loss: 0.5079 - val_acc: 0.9375
Epoch 75/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4358 - acc:
0.9844 - val_loss: 0.5030 - val_acc: 0.9688
Epoch 76/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4246 - acc:
0.9922 - val_loss: 0.4955 - val_acc: 0.9688
Epoch 77/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4111 - acc:
1.0000 - val_loss: 0.5028 - val_acc: 0.9688
Epoch 78/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4062 - acc:
1.0000 - val_loss: 0.5160 - val_acc: 0.9688
Epoch 79/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4119 - acc:
0.9844 - val_loss: 0.4936 - val_acc: 0.9688
Epoch 80/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4083 - acc:
1.0000 - val_loss: 0.4984 - val_acc: 0.9688
Epoch 81/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4036 - acc:
1.0000 - val_loss: 0.5187 - val_acc: 0.9688
Epoch 82/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3936 - acc:
1.0000 - val_loss: 0.5600 - val_acc: 0.9375
Epoch 83/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.4047 - acc:
0.9922 - val_loss: 0.5452 - val_acc: 0.9688
Epoch 84/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3929 - acc:
0.9922 - val_loss: 0.5028 - val_acc: 0.9688
Epoch 85/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3896 - acc:
1.0000 - val_loss: 0.4956 - val_acc: 0.9688
Epoch 86/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3886 - acc:
1.0000 - val_loss: 0.4922 - val_acc: 0.9688
Epoch 87/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3922 - acc:
0.9922 - val_loss: 0.5019 - val_acc: 0.9688
Epoch 88/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3816 - acc:
1.0000 - val_loss: 0.4901 - val_acc: 0.9688
Epoch 89/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3830 - acc:
1.0000 - val_loss: 0.4887 - val_acc: 0.9688
```

```
Epoch 90/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3832 - acc:
1.0000 - val_loss: 0.5063 - val_acc: 0.9688
Epoch 91/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3750 - acc:
1.0000 - val_loss: 0.4945 - val_acc: 0.9688
Epoch 92/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3770 - acc:
1.0000 - val_loss: 0.4959 - val_acc: 0.9688
Epoch 93/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3739 - acc:
1.0000 - val_loss: 0.5152 - val_acc: 0.9688
Epoch 94/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3725 - acc:
1.0000 - val_loss: 0.5663 - val_acc: 0.9375
Epoch 95/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3674 - acc:
1.0000 - val_loss: 0.4963 - val_acc: 0.9688
Epoch 96/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3659 - acc:
1.0000 - val_loss: 0.4822 - val_acc: 0.9688
Epoch 97/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3682 - acc:
1.0000 - val_loss: 0.4995 - val_acc: 0.9688
Epoch 98/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3606 - acc:
1.0000 - val_loss: 0.5633 - val_acc: 0.9375
Epoch 99/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3682 - acc:
0.9922 - val_loss: 0.4856 - val_acc: 0.9688
Epoch 100/100
128/128 [==============================] - 0s 2ms/sample - loss: 0.3594 - acc:
1.0000 - val_loss: 0.4655 - val_acc: 0.9688
```

[24]:

```
# evaluate CNN model3
test_loss_cnn, test_acc_cnn = model_cnn3.evaluate(test_data, test_labels)

print('Test accuracy:', test_acc_cnn)
print('Test loss:', test_loss_cnn)
```

```
40/40 [==============================] - 0s 1ms/sample - loss: 0.5118 - acc:
0.9250
Test accuracy: 0.925
Test loss: 0.5117549538612366
```

[25]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

[26]:

```python
## Data Augmentation

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
↪Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import regularizers
from tensorflow.keras.callbacks import EarlyStopping

train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=20,␣
↪zoom_range=0.2, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.
↪2, horizontal_flip=True)
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow(train_data, train_labels, batch_size=32)
val_generator = val_datagen.flow(val_data, val_labels, batch_size=32)
test_generator = test_datagen.flow(test_data, test_labels, batch_size=32)

model_cnn4 = Sequential()
model_cnn4.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3),␣
↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn4.add(MaxPooling2D((2, 2)))
model_cnn4.add(Conv2D(64, (3, 3), activation='relu',␣
↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn4.add(MaxPooling2D((2, 2)))
model_cnn4.add(Conv2D(128, (3, 3), activation='relu',␣
↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn4.add(MaxPooling2D((2, 2)))
model_cnn4.add(Conv2D(128, (3, 3), activation='relu',␣
↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn4.add(MaxPooling2D((2, 2)))
model_cnn4.add(Flatten())
model_cnn4.add(Dense(512, activation='relu', kernel_regularizer=regularizers.
↪l2(0.001)))
model_cnn4.add(Dropout(0.5))
model_cnn4.add(Dense(1, activation='sigmoid'))

opt = Adam(lr=0.0001)

model_cnn4.compile(optimizer=opt, loss='binary_crossentropy',␣
↪metrics=['accuracy'])

es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=5)
```

```
history_cnn4 = model_cnn4.fit(train_generator, steps_per_epoch=len(train_data)//
↪32, epochs=100, validation_data=val_generator,␣
↪validation_steps=len(val_data)//32, callbacks=[es])
```

```
Epoch 1/100
4/4 [==============================] - 5s 1s/step - loss: 1.9282 - acc: 0.4531 -
val_loss: 1.8876 - val_acc: 0.5312
Epoch 2/100
4/4 [==============================] - 1s 135ms/step - loss: 1.8635 - acc:
0.5000 - val_loss: 1.8234 - val_acc: 0.5312
Epoch 3/100
4/4 [==============================] - 1s 267ms/step - loss: 1.8000 - acc:
0.5000 - val_loss: 1.7611 - val_acc: 0.5312
Epoch 4/100
4/4 [==============================] - 1s 264ms/step - loss: 1.7387 - acc:
0.5000 - val_loss: 1.7016 - val_acc: 0.5312
Epoch 5/100
4/4 [==============================] - 1s 259ms/step - loss: 1.6801 - acc:
0.5000 - val_loss: 1.6448 - val_acc: 0.5312
Epoch 6/100
4/4 [==============================] - 1s 243ms/step - loss: 1.6244 - acc:
0.4922 - val_loss: 1.5908 - val_acc: 0.5312
Epoch 7/100
4/4 [==============================] - 1s 268ms/step - loss: 1.5714 - acc:
0.5000 - val_loss: 1.5397 - val_acc: 0.5312
Epoch 8/100
4/4 [==============================] - 1s 255ms/step - loss: 1.5214 - acc:
0.5000 - val_loss: 1.4914 - val_acc: 0.5312
Epoch 9/100
4/4 [==============================] - 1s 266ms/step - loss: 1.4742 - acc:
0.5000 - val_loss: 1.4459 - val_acc: 0.5312
Epoch 10/100
4/4 [==============================] - 1s 248ms/step - loss: 1.4296 - acc:
0.4922 - val_loss: 1.4030 - val_acc: 0.5312
Epoch 11/100
4/4 [==============================] - 1s 271ms/step - loss: 1.3877 - acc:
0.5078 - val_loss: 1.3626 - val_acc: 0.5312
Epoch 12/100
4/4 [==============================] - 1s 245ms/step - loss: 1.3483 - acc:
0.5000 - val_loss: 1.3246 - val_acc: 0.5312
Epoch 13/100
4/4 [==============================] - 1s 263ms/step - loss: 1.3110 - acc:
0.5078 - val_loss: 1.2889 - val_acc: 0.5312
Epoch 14/100
4/4 [==============================] - 1s 244ms/step - loss: 1.2762 - acc:
0.5156 - val_loss: 1.2554 - val_acc: 0.5312
Epoch 15/100
```

```
4/4 [==============================] - 1s 251ms/step - loss: 1.2436 - acc:
0.5000 - val_loss: 1.2240 - val_acc: 0.5312
Epoch 16/100
4/4 [==============================] - 1s 245ms/step - loss: 1.2129 - acc:
0.5078 - val_loss: 1.1945 - val_acc: 0.5312
Epoch 17/100
4/4 [==============================] - 1s 257ms/step - loss: 1.1840 - acc:
0.5000 - val_loss: 1.1668 - val_acc: 0.5312
Epoch 18/100
4/4 [==============================] - 1s 299ms/step - loss: 1.1570 - acc:
0.4844 - val_loss: 1.1408 - val_acc: 0.5312
Epoch 19/100
4/4 [==============================] - 1s 263ms/step - loss: 1.1315 - acc:
0.5078 - val_loss: 1.1164 - val_acc: 0.5312
Epoch 20/100
4/4 [==============================] - 1s 259ms/step - loss: 1.1079 - acc:
0.5000 - val_loss: 1.0936 - val_acc: 0.5312
Epoch 21/100
4/4 [==============================] - 1s 249ms/step - loss: 1.0855 - acc:
0.5078 - val_loss: 1.0721 - val_acc: 0.5312
Epoch 22/100
4/4 [==============================] - 1s 246ms/step - loss: 1.0644 - acc:
0.5000 - val_loss: 1.0520 - val_acc: 0.5312
Epoch 23/100
4/4 [==============================] - 1s 244ms/step - loss: 1.0451 - acc:
0.5000 - val_loss: 1.0332 - val_acc: 0.5312
Epoch 24/100
4/4 [==============================] - 1s 244ms/step - loss: 1.0264 - acc:
0.5000 - val_loss: 1.0155 - val_acc: 0.5312
Epoch 25/100
4/4 [==============================] - 1s 240ms/step - loss: 1.0092 - acc:
0.5078 - val_loss: 0.9990 - val_acc: 0.5312
Epoch 26/100
4/4 [==============================] - 1s 246ms/step - loss: 0.9931 - acc:
0.4922 - val_loss: 0.9834 - val_acc: 0.5312
Epoch 27/100
4/4 [==============================] - 1s 247ms/step - loss: 0.9782 - acc:
0.4844 - val_loss: 0.9689 - val_acc: 0.5312
Epoch 28/100
4/4 [==============================] - 1s 292ms/step - loss: 0.9639 - acc:
0.4844 - val_loss: 0.9553 - val_acc: 0.5312
Epoch 29/100
4/4 [==============================] - 1s 295ms/step - loss: 0.9504 - acc:
0.5156 - val_loss: 0.9425 - val_acc: 0.5312
Epoch 30/100
4/4 [==============================] - 1s 280ms/step - loss: 0.9380 - acc:
0.5234 - val_loss: 0.9305 - val_acc: 0.5312
Epoch 31/100
```

```
4/4 [==============================] - 1s 279ms/step - loss: 0.9264 - acc:
0.4922 - val_loss: 0.9192 - val_acc: 0.5312
Epoch 32/100
4/4 [==============================] - 1s 308ms/step - loss: 0.9154 - acc:
0.5078 - val_loss: 0.9087 - val_acc: 0.5312
Epoch 33/100
4/4 [==============================] - 1s 268ms/step - loss: 0.9050 - acc:
0.5234 - val_loss: 0.8988 - val_acc: 0.5312
Epoch 34/100
4/4 [==============================] - 1s 245ms/step - loss: 0.8953 - acc:
0.5156 - val_loss: 0.8895 - val_acc: 0.5312
Epoch 35/100
4/4 [==============================] - 1s 241ms/step - loss: 0.8862 - acc:
0.5000 - val_loss: 0.8808 - val_acc: 0.5312
Epoch 36/100
4/4 [==============================] - 1s 240ms/step - loss: 0.8777 - acc:
0.5078 - val_loss: 0.8726 - val_acc: 0.5312
Epoch 37/100
4/4 [==============================] - 1s 259ms/step - loss: 0.8697 - acc:
0.4922 - val_loss: 0.8649 - val_acc: 0.5312
Epoch 38/100
4/4 [==============================] - 1s 259ms/step - loss: 0.8621 - acc:
0.5234 - val_loss: 0.8577 - val_acc: 0.5312
Epoch 39/100
4/4 [==============================] - 1s 253ms/step - loss: 0.8551 - acc:
0.5547 - val_loss: 0.8509 - val_acc: 0.5312
Epoch 40/100
4/4 [==============================] - 1s 269ms/step - loss: 0.8484 - acc:
0.5703 - val_loss: 0.8445 - val_acc: 0.5312
Epoch 41/100
4/4 [==============================] - 1s 268ms/step - loss: 0.8421 - acc:
0.5781 - val_loss: 0.8385 - val_acc: 0.5312
Epoch 42/100
4/4 [==============================] - 1s 338ms/step - loss: 0.8362 - acc:
0.5547 - val_loss: 0.8328 - val_acc: 0.5312
Epoch 43/100
4/4 [==============================] - 1s 333ms/step - loss: 0.8309 - acc:
0.5391 - val_loss: 0.8275 - val_acc: 0.5312
Epoch 44/100
4/4 [==============================] - 1s 259ms/step - loss: 0.8257 - acc:
0.5078 - val_loss: 0.8224 - val_acc: 0.5312
Epoch 45/100
4/4 [==============================] - 1s 251ms/step - loss: 0.8206 - acc:
0.5625 - val_loss: 0.8177 - val_acc: 0.5312
Epoch 46/100
4/4 [==============================] - 1s 265ms/step - loss: 0.8161 - acc:
0.5781 - val_loss: 0.8132 - val_acc: 0.5312
Epoch 47/100
```

```
4/4 [==============================] - 1s 250ms/step - loss: 0.8117 - acc:
0.5703 - val_loss: 0.8090 - val_acc: 0.5312
Epoch 48/100
4/4 [==============================] - 1s 300ms/step - loss: 0.8073 - acc:
0.5703 - val_loss: 0.8050 - val_acc: 0.5312
Epoch 49/100
4/4 [==============================] - 1s 267ms/step - loss: 0.8035 - acc:
0.5859 - val_loss: 0.8012 - val_acc: 0.5312
Epoch 50/100
4/4 [==============================] - 1s 308ms/step - loss: 0.7998 - acc:
0.5234 - val_loss: 0.7976 - val_acc: 0.5312
Epoch 51/100
4/4 [==============================] - 1s 273ms/step - loss: 0.7963 - acc:
0.5781 - val_loss: 0.7942 - val_acc: 0.5312
Epoch 52/100
4/4 [==============================] - 1s 326ms/step - loss: 0.7930 - acc:
0.5312 - val_loss: 0.7910 - val_acc: 0.5312
Epoch 53/100
4/4 [==============================] - 1s 253ms/step - loss: 0.7897 - acc:
0.6016 - val_loss: 0.7879 - val_acc: 0.5312
Epoch 54/100
4/4 [==============================] - 1s 277ms/step - loss: 0.7868 - acc:
0.6016 - val_loss: 0.7850 - val_acc: 0.6250
Epoch 55/100
4/4 [==============================] - 1s 277ms/step - loss: 0.7840 - acc:
0.5781 - val_loss: 0.7821 - val_acc: 0.5312
Epoch 56/100
4/4 [==============================] - 1s 277ms/step - loss: 0.7812 - acc:
0.5391 - val_loss: 0.7795 - val_acc: 0.5312
Epoch 57/100
4/4 [==============================] - 1s 258ms/step - loss: 0.7784 - acc:
0.5781 - val_loss: 0.7770 - val_acc: 0.5312
Epoch 58/100
4/4 [==============================] - 1s 262ms/step - loss: 0.7760 - acc:
0.6406 - val_loss: 0.7745 - val_acc: 0.5312
Epoch 59/100
4/4 [==============================] - 1s 248ms/step - loss: 0.7737 - acc:
0.6328 - val_loss: 0.7722 - val_acc: 0.6250
Epoch 60/100
4/4 [==============================] - 1s 275ms/step - loss: 0.7712 - acc:
0.6562 - val_loss: 0.7700 - val_acc: 0.6250
Epoch 61/100
4/4 [==============================] - 1s 271ms/step - loss: 0.7691 - acc:
0.6719 - val_loss: 0.7678 - val_acc: 0.6250
Epoch 62/100
4/4 [==============================] - 1s 264ms/step - loss: 0.7669 - acc:
0.6875 - val_loss: 0.7657 - val_acc: 0.7812
Epoch 63/100
```

```
4/4 [==============================] - 1s 256ms/step - loss: 0.7647 - acc:
0.6953 - val_loss: 0.7636 - val_acc: 0.6562
Epoch 64/100
4/4 [==============================] - 1s 274ms/step - loss: 0.7627 - acc:
0.6406 - val_loss: 0.7616 - val_acc: 0.6562
Epoch 65/100
4/4 [==============================] - 1s 272ms/step - loss: 0.7606 - acc:
0.7578 - val_loss: 0.7595 - val_acc: 0.7812
Epoch 66/100
4/4 [==============================] - 1s 302ms/step - loss: 0.7582 - acc:
0.7031 - val_loss: 0.7573 - val_acc: 0.6875
Epoch 67/100
4/4 [==============================] - 1s 289ms/step - loss: 0.7558 - acc:
0.8281 - val_loss: 0.7551 - val_acc: 0.7812
Epoch 68/100
4/4 [==============================] - 1s 266ms/step - loss: 0.7539 - acc:
0.6953 - val_loss: 0.7521 - val_acc: 0.6562
Epoch 69/100
4/4 [==============================] - 1s 295ms/step - loss: 0.7496 - acc:
0.7812 - val_loss: 0.7490 - val_acc: 0.7812
Epoch 70/100
4/4 [==============================] - 1s 280ms/step - loss: 0.7450 - acc:
0.8438 - val_loss: 0.7437 - val_acc: 0.7812
Epoch 71/100
4/4 [==============================] - 1s 245ms/step - loss: 0.7392 - acc:
0.7969 - val_loss: 0.7370 - val_acc: 0.8125
Epoch 72/100
4/4 [==============================] - 1s 244ms/step - loss: 0.7257 - acc:
0.8359 - val_loss: 0.7247 - val_acc: 0.8125
Epoch 73/100
4/4 [==============================] - 1s 284ms/step - loss: 0.7088 - acc:
0.8594 - val_loss: 0.7117 - val_acc: 0.7500
Epoch 74/100
4/4 [==============================] - 1s 261ms/step - loss: 0.6863 - acc:
0.8203 - val_loss: 0.6755 - val_acc: 0.8125
Epoch 75/100
4/4 [==============================] - 1s 256ms/step - loss: 0.6462 - acc:
0.8828 - val_loss: 0.6387 - val_acc: 0.7812
Epoch 76/100
4/4 [==============================] - 1s 274ms/step - loss: 0.5858 - acc:
0.8750 - val_loss: 0.6203 - val_acc: 0.7500
Epoch 77/100
4/4 [==============================] - 1s 263ms/step - loss: 0.5346 - acc:
0.8438 - val_loss: 0.5448 - val_acc: 0.8125
Epoch 78/100
4/4 [==============================] - 1s 269ms/step - loss: 0.4545 - acc:
0.8594 - val_loss: 0.5165 - val_acc: 0.7812
Epoch 79/100
```

```
4/4 [==============================] - 1s 272ms/step - loss: 0.4208 - acc:
0.8594 - val_loss: 0.5347 - val_acc: 0.8438
Epoch 80/100
4/4 [==============================] - 1s 264ms/step - loss: 0.3968 - acc:
0.8906 - val_loss: 0.5041 - val_acc: 0.8125
Epoch 81/100
4/4 [==============================] - 1s 250ms/step - loss: 0.3798 - acc:
0.8906 - val_loss: 0.4760 - val_acc: 0.8125
Epoch 82/100
4/4 [==============================] - 1s 245ms/step - loss: 0.4405 - acc:
0.8516 - val_loss: 0.5018 - val_acc: 0.7812
Epoch 83/100
4/4 [==============================] - 1s 269ms/step - loss: 0.4498 - acc:
0.8594 - val_loss: 0.4680 - val_acc: 0.8750
Epoch 84/100
4/4 [==============================] - 1s 249ms/step - loss: 0.3274 - acc:
0.8906 - val_loss: 0.4901 - val_acc: 0.8750
Epoch 85/100
4/4 [==============================] - 1s 241ms/step - loss: 0.3747 - acc:
0.8828 - val_loss: 0.4803 - val_acc: 0.8438
Epoch 86/100
4/4 [==============================] - 1s 248ms/step - loss: 0.4111 - acc:
0.8906 - val_loss: 0.4797 - val_acc: 0.8750
Epoch 87/100
4/4 [==============================] - 1s 243ms/step - loss: 0.3512 - acc:
0.9219 - val_loss: 0.5105 - val_acc: 0.8438
Epoch 88/100
4/4 [==============================] - 1s 245ms/step - loss: 0.4507 - acc:
0.8906 - val_loss: 0.4664 - val_acc: 0.8750
Epoch 89/100
4/4 [==============================] - 1s 237ms/step - loss: 0.4495 - acc:
0.8281 - val_loss: 0.4547 - val_acc: 0.8438
Epoch 90/100
4/4 [==============================] - 1s 245ms/step - loss: 0.3596 - acc:
0.9219 - val_loss: 0.5059 - val_acc: 0.8125
Epoch 91/100
4/4 [==============================] - 1s 245ms/step - loss: 0.3781 - acc:
0.8906 - val_loss: 0.5382 - val_acc: 0.8125
Epoch 92/100
4/4 [==============================] - 1s 251ms/step - loss: 0.3932 - acc:
0.8594 - val_loss: 0.5286 - val_acc: 0.8125
Epoch 93/100
4/4 [==============================] - 1s 247ms/step - loss: 0.3279 - acc:
0.9062 - val_loss: 0.5189 - val_acc: 0.8125
Epoch 94/100
4/4 [==============================] - 1s 241ms/step - loss: 0.3256 - acc:
0.8984 - val_loss: 0.5201 - val_acc: 0.8125
Epoch 00094: early stopping
```

[27]:
```python
test_loss, test_acc = model_cnn4.evaluate_generator(test_generator)

print('Test accuracy:', test_acc)
print('Test loss:', test_loss)
```

```
Test accuracy: 0.925
Test loss: 0.45434319972991943
```

In this code, we first define the CNN model architecture, which consists of two fully linked layers and four convolutional layers. The model is built utilising accuracy and binary cross-entropy loss as evaluation metrics.

The ImageDataGenerator function from Keras is then used for data augmentation to create new images by randomly applying transformations like rotation, zoom, shift, and flip to the source images. This method aids in expanding the training set's size and guards against overfitting.

Additionally, if the validation loss does not decrease after a predetermined number of epochs, we define an EarlyStopping callback to end training. By doing so, generalisation is enhanced and overfitting is prevented.

Finally, we fit the CNN model using the fit_generator function from Keras with the training data, training labels, validation data, and validation labels. We use a batch size of 32 and train the model for a maximum of 100 epochs.

Now, let's develop a FNN model for the same classification task.

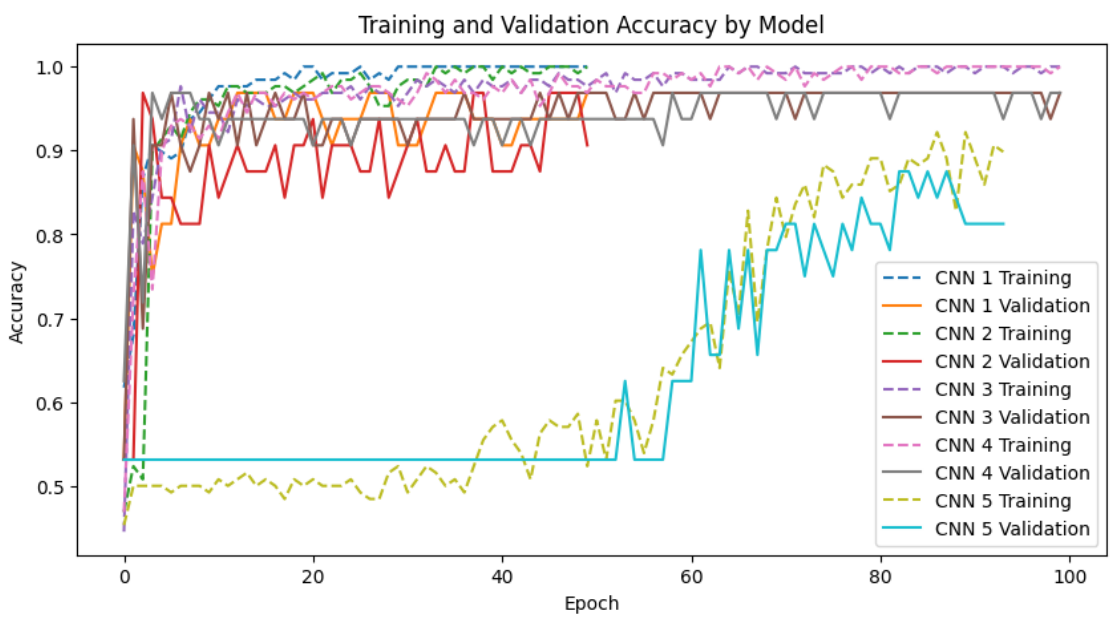
[28]:
```python
history_cnn.history.keys()
```

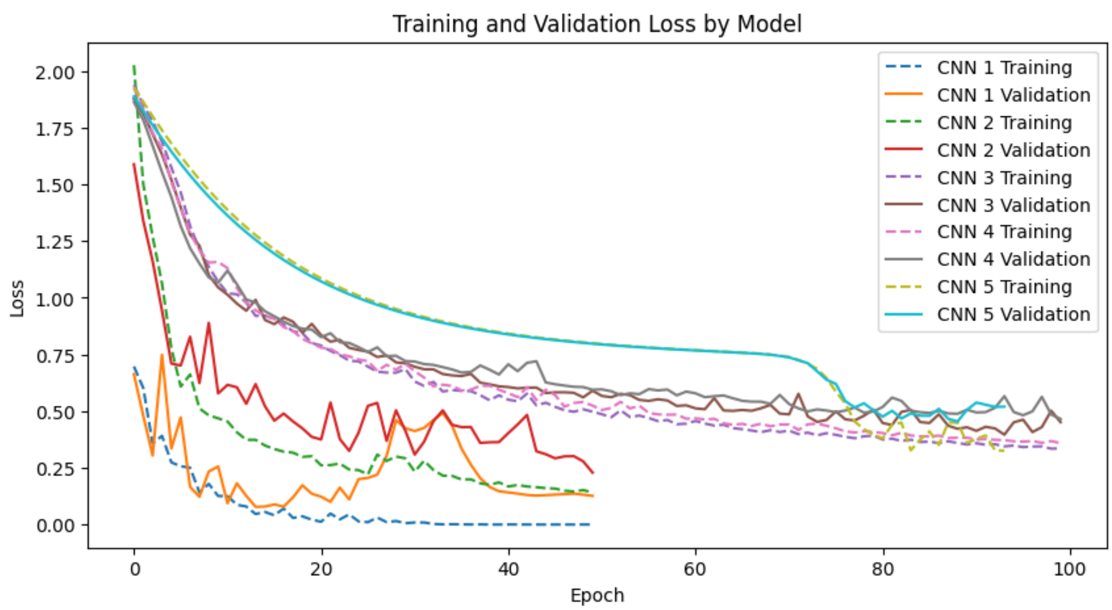
[28]:
```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

[29]:
```python
import matplotlib.pyplot as plt
```

[30]:
```python
# Plot the training and validation accuracy
plt.figure(figsize=(10, 5))
plt.plot(history_cnn.history['acc'], label='CNN 1 Training', linestyle='--')
plt.plot(history_cnn.history['val_acc'], label='CNN 1 Validation')
plt.plot(history_cnn1.history['acc'], label='CNN 2 Training', linestyle='--')
plt.plot(history_cnn1.history['val_acc'], label='CNN 2 Validation')
plt.plot(history_cnn2.history['acc'], label='CNN 3 Training', linestyle='--')
plt.plot(history_cnn2.history['val_acc'], label='CNN 3 Validation')
plt.plot(history_cnn3.history['acc'], label='CNN 4 Training', linestyle='--')
plt.plot(history_cnn3.history['val_acc'], label='CNN 4 Validation')
plt.plot(history_cnn4.history['acc'], label='CNN 5 Training', linestyle='--')
plt.plot(history_cnn4.history['val_acc'], label='CNN 5 Validation')
plt.title('Training and Validation Accuracy by Model')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.show()
```

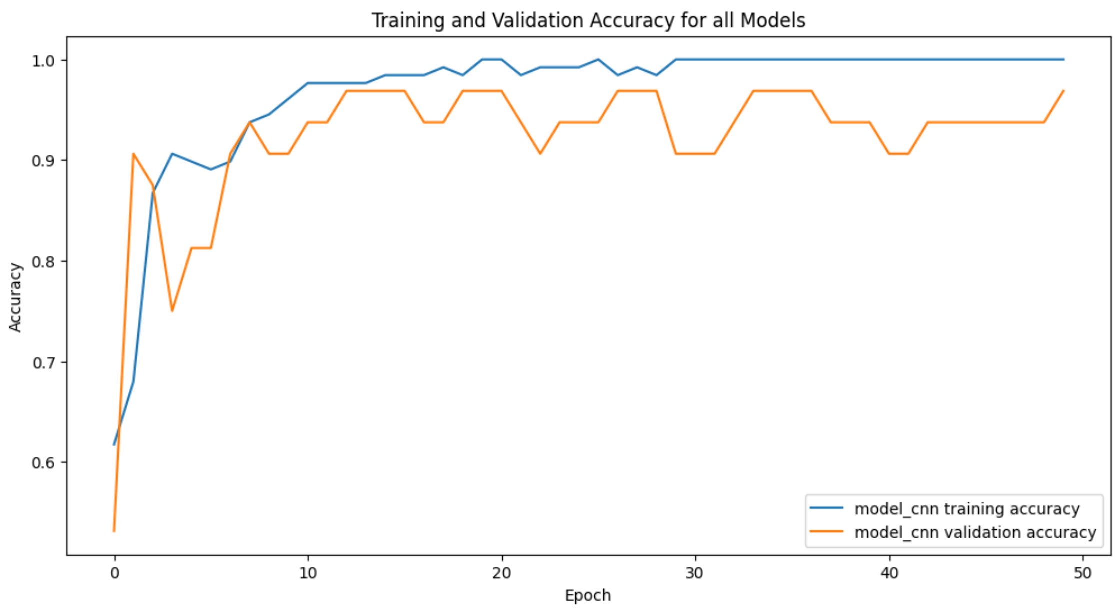
```python
# Plot the training and validation loss
plt.figure(figsize=(10, 5))
plt.plot(history_cnn.history['loss'], label='CNN 1 Training', linestyle='--')
plt.plot(history_cnn.history['val_loss'], label='CNN 1 Validation')
plt.plot(history_cnn1.history['loss'], label='CNN 2 Training', linestyle='--')
plt.plot(history_cnn1.history['val_loss'], label='CNN 2 Validation')
plt.plot(history_cnn2.history['loss'], label='CNN 3 Training', linestyle='--')
plt.plot(history_cnn2.history['val_loss'], label='CNN 3 Validation')
plt.plot(history_cnn3.history['loss'], label='CNN 4 Training', linestyle='--')
plt.plot(history_cnn3.history['val_loss'], label='CNN 4 Validation')
plt.plot(history_cnn4.history['loss'], label='CNN 5 Training', linestyle='--')
plt.plot(history_cnn4.history['val_loss'], label='CNN 5 Validation')
plt.title('Training and Validation Loss by Model')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.show()
```

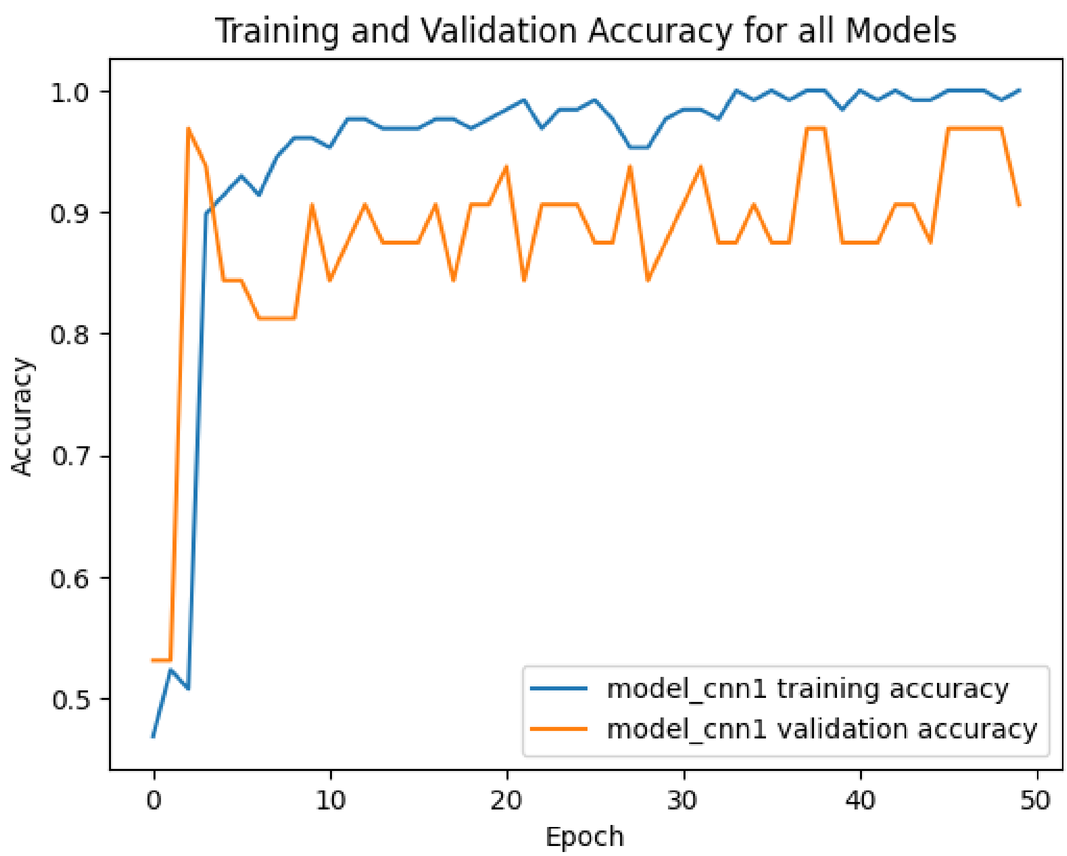


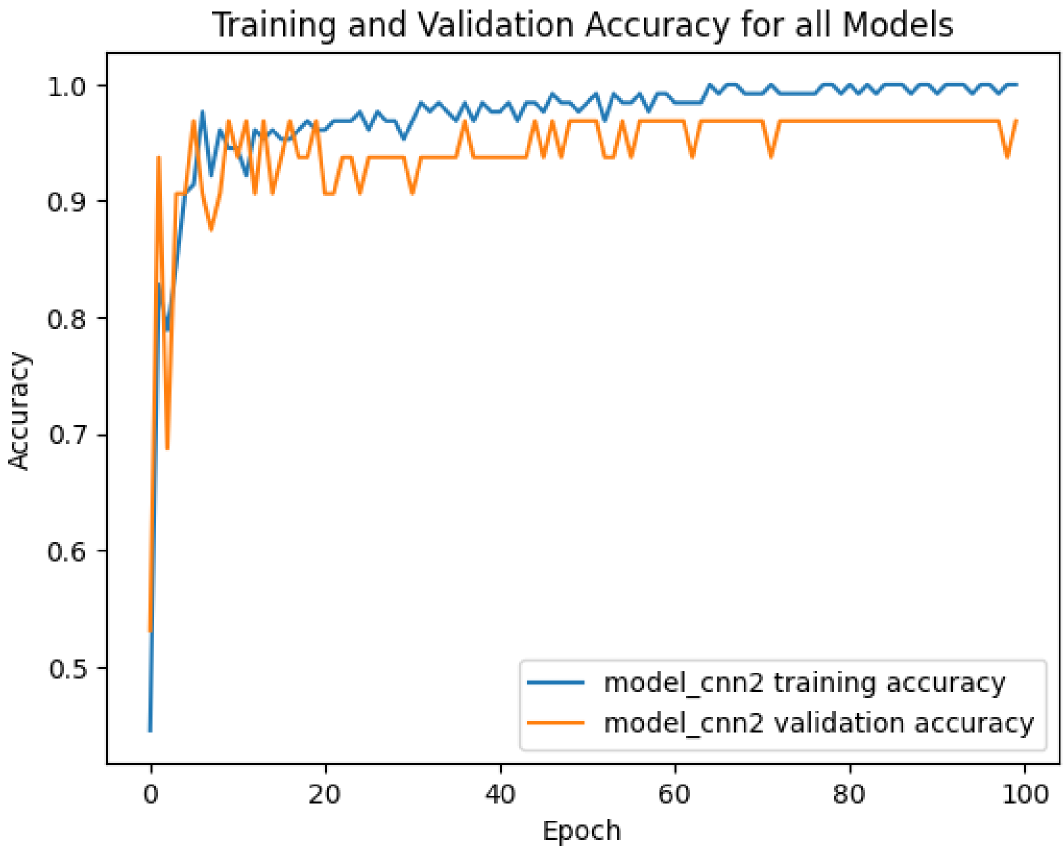


[31]:
```
# Create a table comparing the accuracy and loss of each model on the test set
test_acc = []
test_loss = []

test_acc.append(model_cnn.evaluate(test_data, test_labels, verbose=0)[1])
test_loss.append(model_cnn.evaluate(test_data, test_labels, verbose=0)[0])

# evaluate model_cnn1 on test data
test_acc.append(model_cnn1.evaluate(test_data, test_labels, verbose=0)[1])
test_loss.append(model_cnn1.evaluate(test_data, test_labels, verbose=0)[0])

# evaluate model_cnn2 on test data
test_acc.append(model_cnn2.evaluate(test_data, test_labels, verbose=0)[1])
test_loss.append(model_cnn2.evaluate(test_data, test_labels, verbose=0)[0])

# evaluate model_cnn3 on test data
test_acc.append(model_cnn3.evaluate(test_data, test_labels, verbose=0)[1])
test_loss.append(model_cnn3.evaluate(test_data, test_labels, verbose=0)[0])

# evaluate model_cnn4 on test data
test_acc.append(model_cnn4.evaluate(test_data, test_labels, verbose=0)[1])
test_loss.append(model_cnn4.evaluate(test_data, test_labels, verbose=0)[0])

# create a dictionary to store training and testing metrics for each model
metrics = {'model_cnn': {'train_acc': history_cnn.history['acc'], 'train_loss':␣
↪history_cnn.history['loss'],
```

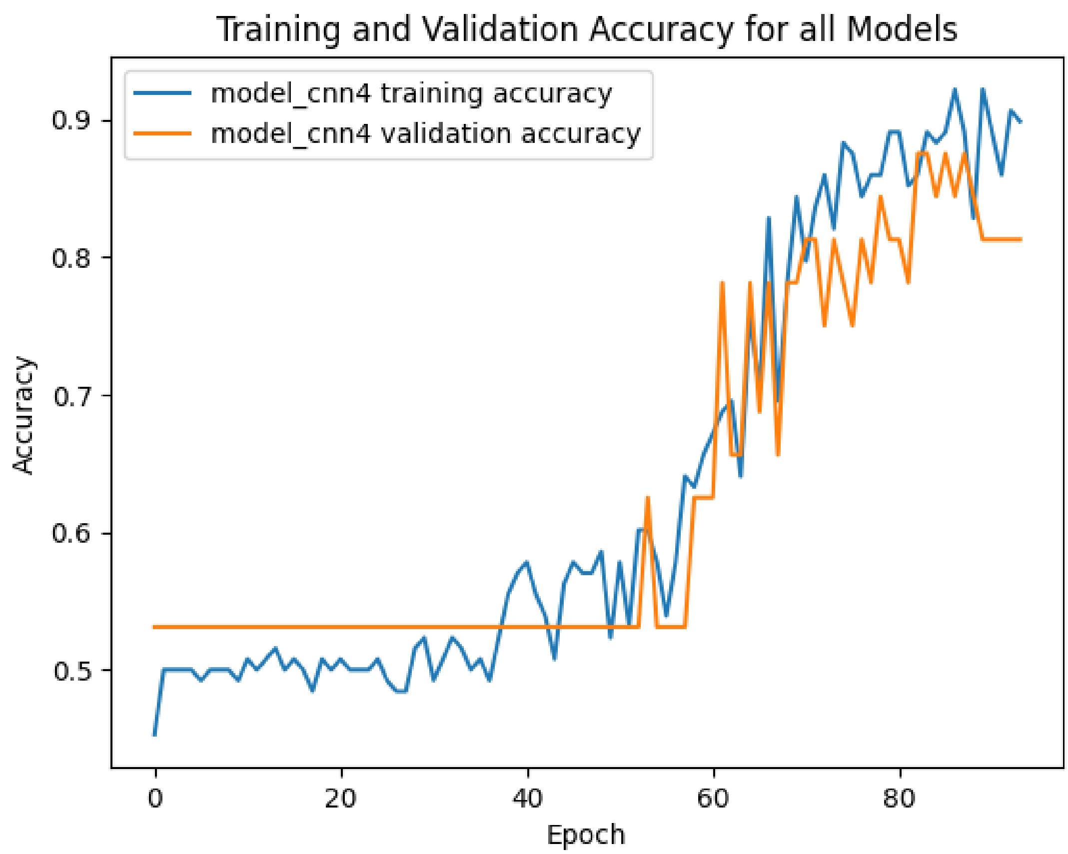
```
'val_acc': history_cnn.history['val_acc'], 'val_loss': history_cnn.
 ↪history['val_loss'],
'test_acc': test_acc[0], 'test_loss': test_loss[0]},
'model_cnn1': {'train_acc': history_cnn1.history['acc'], 'train_loss':␣
 ↪history_cnn1.history['loss'],
'val_acc': history_cnn1.history['val_acc'], 'val_loss': history_cnn1.
 ↪history['val_loss'],
'test_acc': test_acc[1], 'test_loss': test_loss[1]},
'model_cnn2': {'train_acc': history_cnn2.history['acc'], 'train_loss':␣
 ↪history_cnn2.history['loss'],
'val_acc': history_cnn2.history['val_acc'], 'val_loss': history_cnn2.
 ↪history['val_loss'],
'test_acc': test_acc[2], 'test_loss': test_loss[2]},
'model_cnn3': {'train_acc': history_cnn3.history['acc'], 'train_loss':␣
 ↪history_cnn3.history['loss'],
'val_acc': history_cnn3.history['val_acc'], 'val_loss': history_cnn3.
 ↪history['val_loss'],
'test_acc': test_acc[3], 'test_loss': test_loss[3]},
'model_cnn4': {'train_acc': history_cnn4.history['acc'], 'train_loss':␣
 ↪history_cnn4.history['loss'],
'val_acc': history_cnn4.history['val_acc'], 'val_loss': history_cnn4.
 ↪history['val_loss'],
'test_acc': test_acc[4], 'test_loss': test_loss[4]}}

# plot training and validation accuracy for all models
plt.figure(figsize=(12, 6))
for model, metrics_dict in metrics.items():
    plt.plot(metrics_dict['train_acc'], label=model+' training accuracy')
    plt.plot(metrics_dict['val_acc'], label=model+' validation accuracy')
    plt.title('Training and Validation Accuracy for all Models')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()

# plot training and validation loss for all models
plt.figure(figsize=(12, 6))
for model, metrics_dict in metrics.items():
    plt.plot(metrics_dict['train_loss'], label=model+' training loss')
    plt.plot(metrics_dict['val_loss'], label=model+' validation loss')
    plt.title('Training and Validation Loss for all Models')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
```

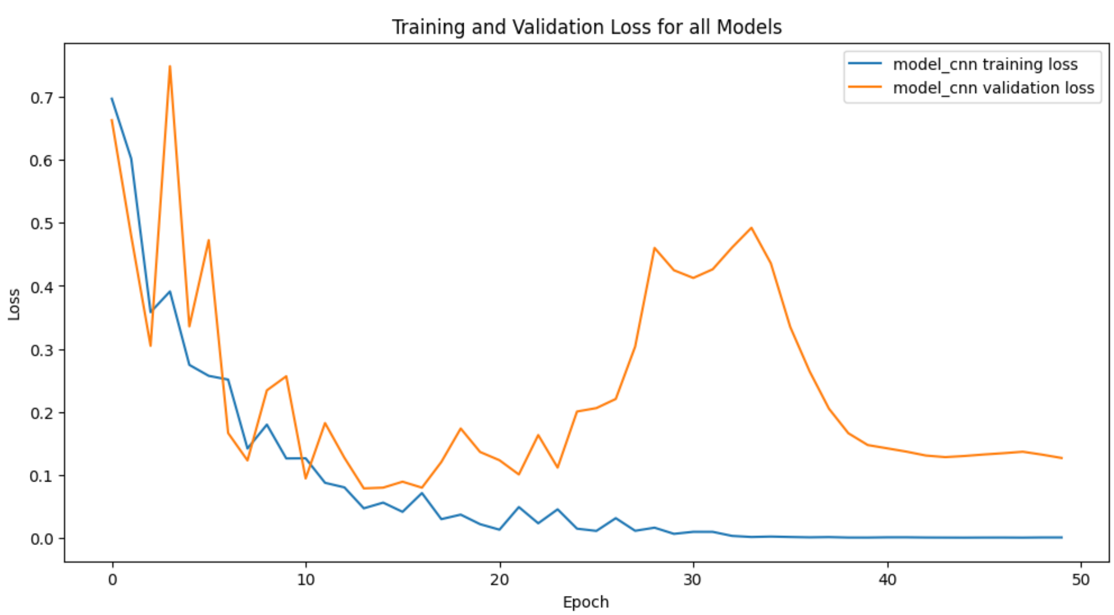

Training and Validation Accuracy for all Models
Accuracy
Epoch
model_cnn training accuracy
model_cnn validation accuracy

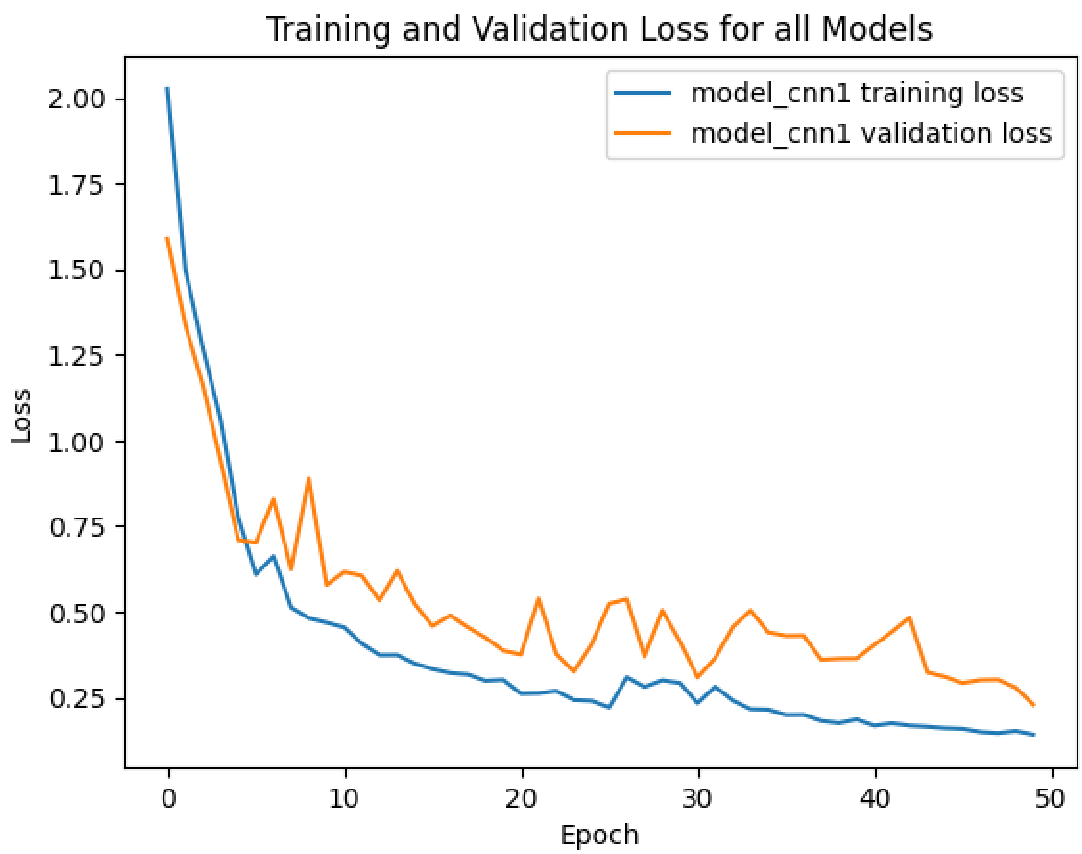

Training and Validation Accuracy for all Models
Accuracy
Epoch
model_cnn1 training accuracy
model_cnn1 validation accuracy

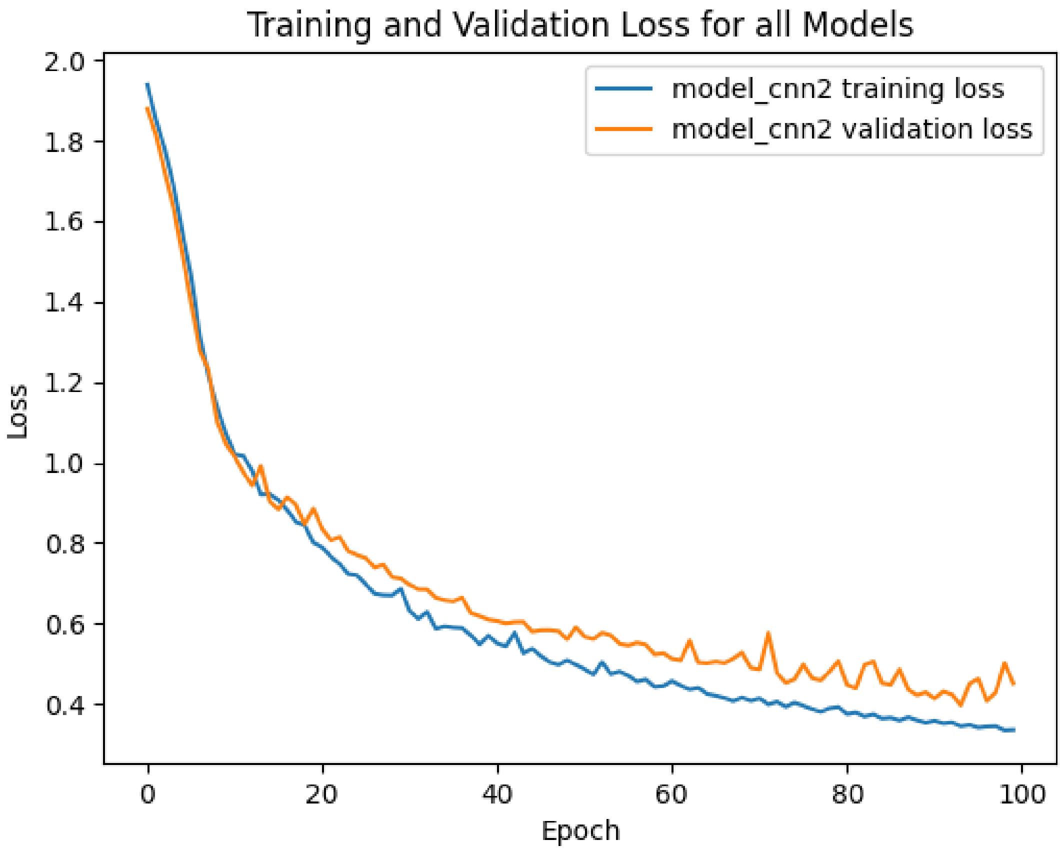

Training and Validation Accuracy for all Models
Accuracy
Epoch
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100
model_cnn2 training accuracy
model_cnn2 validation accuracy

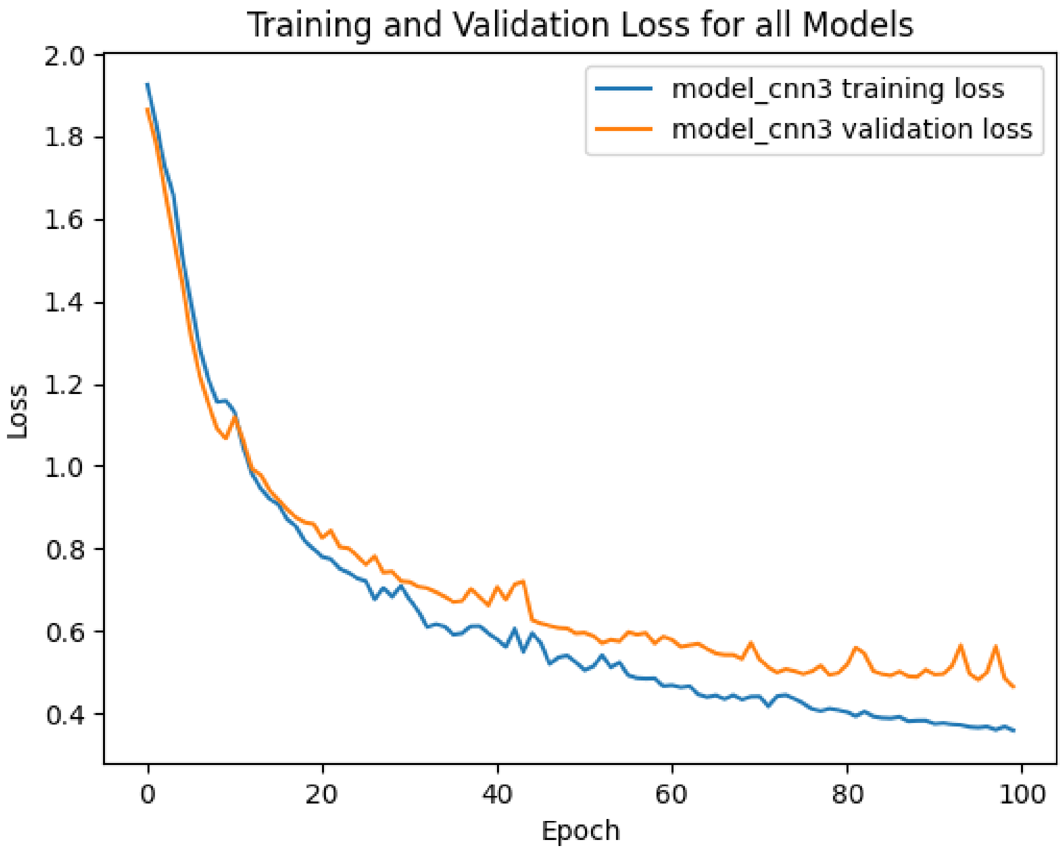

Training and Validation Accuracy for all Models
Accuracy
Epoch
model_cnn3 training accuracy
model_cnn3 validation accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100

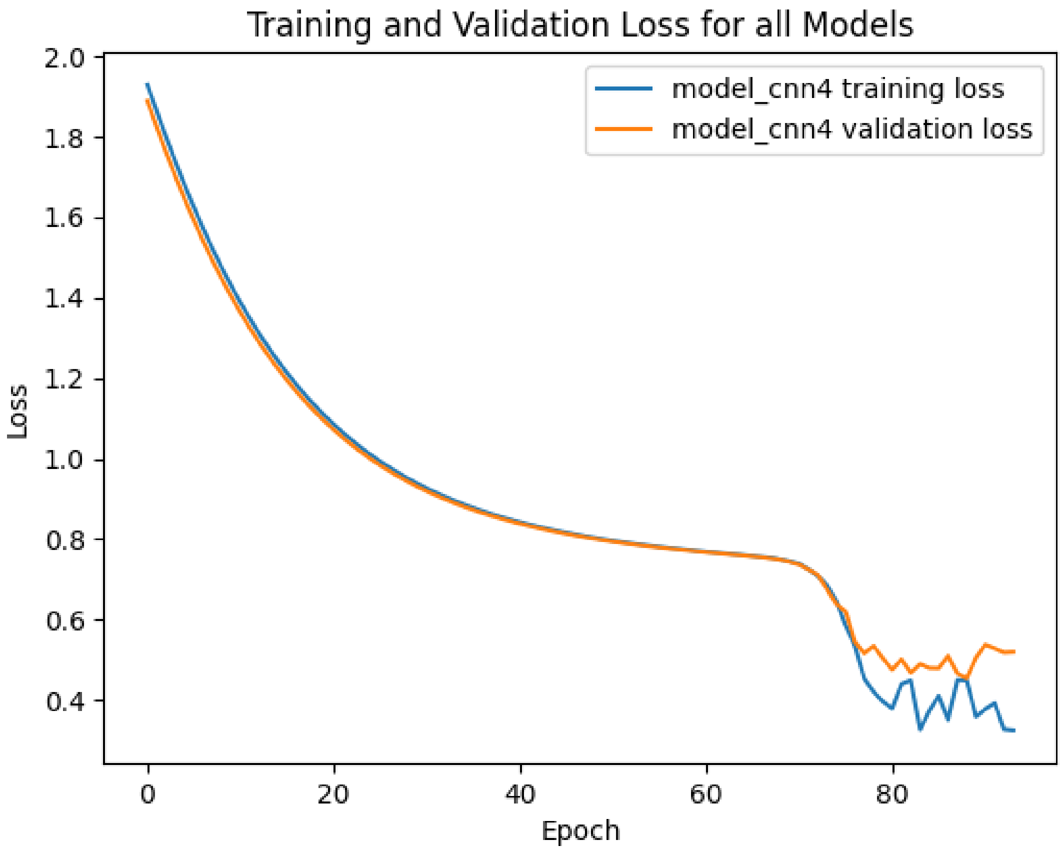

Training and Validation Accuracy for all Models
model_cnn4 training accuracy
model_cnn4 validation accuracy
Accuracy
Epoch
0.5
0.6
0.7
0.8
0.9
0
20
40
60
80


Training and Validation Loss for all Models
model_cnn training loss
model_cnn validation loss
Loss
Epoch
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
10
20
30
40
50


Training and Validation Loss for all Models
model_cnn1 training loss
model_cnn1 validation loss
Loss
Epoch
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
10
20
30
40
50


Training and Validation Loss for all Models
model_cnn2 training loss
model_cnn2 validation loss
Loss
Epoch
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
20
40
60
80
100


Training and Validation Loss for all Models
model_cnn3 training loss
model_cnn3 validation loss
Loss
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
20
40
60
80
100
Epoch



[ ]:
```
train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=20,␣
  ↪zoom_range=0.2, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.
  ↪2, horizontal_flip=True)
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow(train_data, train_labels, batch_size=32)
val_generator = val_datagen.flow(val_data, val_labels, batch_size=32)
test_generator = test_datagen.flow(test_data, test_labels, batch_size=32)

# Define individual models
model_cnn3 = Sequential()
model_cnn3.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3),␣
  ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn3.add(MaxPooling2D((2, 2)))
model_cnn3.add(Conv2D(64, (3, 3), activation='relu',␣
  ↪kernel_regularizer=regularizers.l2(0.001)))
model_cnn3.add(MaxPooling2D((2, 2)))
```

```
model_cnn3.add(Conv2D(128, (3, 3), activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model_cnn3.add(MaxPooling2D((2, 2)))
model_cnn3.add(Conv2D(128, (3, 3), activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model_cnn3.add(MaxPooling2D((2, 2)))
model_cnn3.add(Flatten())
model_cnn3.add(Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model_cnn3.add(Dropout(0.5))
model_cnn3.add(Dense(1, activation='sigmoid'))
opt = Adam(lr=0.0001)
model_cnn3.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=5)
history1 = model_cnn3.fit_generator(train_datagen.flow(train_data, train_labels, batch_size=32), steps_per_epoch=len(train_data)//32, epochs=100, validation_data=val_datagen.flow(val_data, val_labels), validation_steps=len(val_data)//32, callbacks=[es])


# Define individual models
model_cnn4 = Sequential()
model_cnn4.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3), kernel_regularizer=regularizers.l2(0.001)))
model_cnn4.add(MaxPooling2D((2, 2)))
model_cnn4.add(Conv2D(64, (3, 3), activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model_cnn4.add(MaxPooling2D((2, 2)))
model_cnn4.add(Conv2D(128, (3, 3), activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model_cnn4.add(MaxPooling2D((2, 2)))
model_cnn4.add(Conv2D(128, (3, 3), activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model_cnn4.add(MaxPooling2D((2, 2)))
model_cnn4.add(Flatten())
model_cnn4.add(Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model_cnn4.add(Dropout(0.5))
model_cnn4.add(Dense(1, activation='sigmoid'))
opt = Adam(lr=0.0001)
model_cnn4.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
```

```
history2 = model_cnn4.fit_generator(train_datagen.flow(train_data,␣
↪train_labels, batch_size=32), steps_per_epoch=len(train_data)//32,␣
↪epochs=100, validation_data=val_datagen.flow(val_data, val_labels),␣
↪validation_steps=len(val_data)//32, callbacks=[es])


# Predictions of individual models
pred1 = model_cnn3.predict(test_data)
pred2 = model_cnn4.predict(test_data)

# Average predictions of individual models
ensemble_pred = (pred1 + pred2) / 2

# Convert predictions to binary values
ensemble_pred_binary = (ensemble_pred > 0.5).astype('int')

# Evaluate individual models
loss1, acc1 = model_cnn3.evaluate(test_data, test_labels, verbose=0)
loss2, acc2 = model_cnn4.evaluate(test_data, test_labels, verbose=0)

# Evaluate ensemble model
ensemble_loss, ensemble_acc = model_cnn3.evaluate(test_data,␣
↪ensemble_pred_binary, verbose=0)

print('Individual model 1 - Loss: {}, Accuracy: {}'.format(loss1, acc1))
print('Individual model 2 - Loss: {}, Accuracy: {}'.format(loss2, acc2))
print('Ensemble model - Loss: {}, Accuracy: {}'.format(ensemble_loss,␣
↪ensemble_acc))
```

```
WARNING:tensorflow:From c:\Users\abhij\.conda\envs\image_ML2\lib\site-
packages\tensorflow\python\ops\init_ops.py:1251: calling
VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is
deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the
constructor
WARNING:tensorflow:From c:\Users\abhij\.conda\envs\image_ML2\lib\site-
packages\tensorflow\python\ops\nn_impl.py:180:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Epoch 1/100
4/4 [==============================] - 4s 1s/step - loss: 1.9278 - acc: 0.4766 -
val_loss: 1.8873 - val_acc: 0.4688
Epoch 2/100
4/4 [==============================] - 1s 264ms/step - loss: 1.8630 - acc:
```

```
0.4922 - val_loss: 1.8230 - val_acc: 0.4688
Epoch 3/100
4/4 [==============================] - 1s 275ms/step - loss: 1.7994 - acc:
0.5469 - val_loss: 1.7607 - val_acc: 0.4688
Epoch 4/100
4/4 [==============================] - 1s 258ms/step - loss: 1.7381 - acc:
0.5000 - val_loss: 1.7011 - val_acc: 0.4688
Epoch 5/100
4/4 [==============================] - 1s 263ms/step - loss: 1.6796 - acc:
0.5469 - val_loss: 1.6443 - val_acc: 0.6250
Epoch 6/100
4/4 [==============================] - 1s 249ms/step - loss: 1.6238 - acc:
0.5156 - val_loss: 1.5904 - val_acc: 0.4688
Epoch 7/100
4/4 [==============================] - 1s 250ms/step - loss: 1.5710 - acc:
0.5234 - val_loss: 1.5393 - val_acc: 0.5312
Epoch 8/100
4/4 [==============================] - 1s 266ms/step - loss: 1.5210 - acc:
0.5469 - val_loss: 1.4911 - val_acc: 0.5312
Epoch 9/100
4/4 [==============================] - 1s 245ms/step - loss: 1.4739 - acc:
0.5312 - val_loss: 1.4457 - val_acc: 0.5312
Epoch 10/100
4/4 [==============================] - 1s 245ms/step - loss: 1.4294 - acc:
0.5312 - val_loss: 1.4029 - val_acc: 0.5312
Epoch 11/100
4/4 [==============================] - 1s 247ms/step - loss: 1.3875 - acc:
0.5547 - val_loss: 1.3626 - val_acc: 0.5312
Epoch 12/100
4/4 [==============================] - 1s 268ms/step - loss: 1.3482 - acc:
0.5391 - val_loss: 1.3248 - val_acc: 0.6250
Epoch 13/100
4/4 [==============================] - 1s 241ms/step - loss: 1.3112 - acc:
0.5000 - val_loss: 1.2892 - val_acc: 0.6875
Epoch 14/100
4/4 [==============================] - 1s 253ms/step - loss: 1.2765 - acc:
0.5234 - val_loss: 1.2557 - val_acc: 0.5312
Epoch 15/100
4/4 [==============================] - 1s 280ms/step - loss: 1.2439 - acc:
0.5078 - val_loss: 1.2243 - val_acc: 0.5312
Epoch 16/100
4/4 [==============================] - 1s 256ms/step - loss: 1.2131 - acc:
0.5703 - val_loss: 1.1949 - val_acc: 0.5312
Epoch 17/100
4/4 [==============================] - 1s 246ms/step - loss: 1.1843 - acc:
0.5156 - val_loss: 1.1672 - val_acc: 0.5312
Epoch 18/100
4/4 [==============================] - 1s 243ms/step - loss: 1.1573 - acc:
```

```
0.5938 - val_loss: 1.1413 - val_acc: 0.7812
Epoch 19/100
4/4 [==============================] - 1s 247ms/step - loss: 1.1319 - acc:
0.6094 - val_loss: 1.1170 - val_acc: 0.5312
Epoch 20/100
4/4 [==============================] - 1s 244ms/step - loss: 1.1083 - acc:
0.5234 - val_loss: 1.0941 - val_acc: 0.5312
Epoch 21/100
4/4 [==============================] - 1s 241ms/step - loss: 1.0861 - acc:
0.5000 - val_loss: 1.0728 - val_acc: 0.4688
Epoch 22/100
4/4 [==============================] - 1s 245ms/step - loss: 1.0653 - acc:
0.4922 - val_loss: 1.0528 - val_acc: 0.4688
Epoch 23/100
4/4 [==============================] - 1s 249ms/step - loss: 1.0455 - acc:
0.5312 - val_loss: 1.0339 - val_acc: 0.4688
Epoch 24/100
4/4 [==============================] - 1s 243ms/step - loss: 1.0272 - acc:
0.5391 - val_loss: 1.0162 - val_acc: 0.6875
Epoch 25/100
4/4 [==============================] - 1s 240ms/step - loss: 1.0099 - acc:
0.5625 - val_loss: 0.9998 - val_acc: 0.4688
Epoch 26/100
4/4 [==============================] - 1s 248ms/step - loss: 0.9938 - acc:
0.5938 - val_loss: 0.9842 - val_acc: 0.4688
Epoch 27/100
4/4 [==============================] - 1s 245ms/step - loss: 0.9786 - acc:
0.5703 - val_loss: 0.9696 - val_acc: 0.7188
Epoch 28/100
4/4 [==============================] - 1s 246ms/step - loss: 0.9642 - acc:
0.6641 - val_loss: 0.9560 - val_acc: 0.7812
Epoch 29/100
4/4 [==============================] - 1s 245ms/step - loss: 0.9510 - acc:
0.6094 - val_loss: 0.9432 - val_acc: 0.7812
Epoch 30/100
4/4 [==============================] - 1s 244ms/step - loss: 0.9386 - acc:
0.5859 - val_loss: 0.9311 - val_acc: 0.5312
Epoch 31/100
4/4 [==============================] - 1s 243ms/step - loss: 0.9268 - acc:
0.5547 - val_loss: 0.9200 - val_acc: 0.5312
Epoch 32/100
4/4 [==============================] - 1s 240ms/step - loss: 0.9158 - acc:
0.5156 - val_loss: 0.9095 - val_acc: 0.4688
Epoch 33/100
4/4 [==============================] - 1s 244ms/step - loss: 0.9056 - acc:
0.5391 - val_loss: 0.8995 - val_acc: 0.4688
Epoch 34/100
4/4 [==============================] - 1s 241ms/step - loss: 0.8959 - acc:
```

```
0.5312 - val_loss: 0.8902 - val_acc: 0.4688
Epoch 35/100
4/4 [==============================] - 1s 247ms/step - loss: 0.8868 - acc:
0.5234 - val_loss: 0.8815 - val_acc: 0.4688
Epoch 36/100
4/4 [==============================] - 1s 251ms/step - loss: 0.8783 - acc:
0.5625 - val_loss: 0.8733 - val_acc: 0.4688
Epoch 37/100
4/4 [==============================] - 1s 263ms/step - loss: 0.8703 - acc:
0.5391 - val_loss: 0.8656 - val_acc: 0.4688
Epoch 38/100
4/4 [==============================] - 1s 244ms/step - loss: 0.8626 - acc:
0.5547 - val_loss: 0.8584 - val_acc: 0.4688
Epoch 39/100
4/4 [==============================] - 1s 244ms/step - loss: 0.8557 - acc:
0.5391 - val_loss: 0.8515 - val_acc: 0.4688
Epoch 40/100
4/4 [==============================] - 1s 245ms/step - loss: 0.8490 - acc:
0.5781 - val_loss: 0.8451 - val_acc: 0.4688
Epoch 41/100
4/4 [==============================] - 1s 252ms/step - loss: 0.8427 - acc:
0.6250 - val_loss: 0.8390 - val_acc: 0.8125
Epoch 42/100
4/4 [==============================] - 1s 241ms/step - loss: 0.8369 - acc:
0.5859 - val_loss: 0.8333 - val_acc: 0.7812
Epoch 43/100
4/4 [==============================] - 1s 241ms/step - loss: 0.8311 - acc:
0.6406 - val_loss: 0.8279 - val_acc: 0.7812
Epoch 44/100
4/4 [==============================] - 1s 248ms/step - loss: 0.8259 - acc:
0.6875 - val_loss: 0.8229 - val_acc: 0.6562
Epoch 45/100
4/4 [==============================] - 1s 237ms/step - loss: 0.8208 - acc:
0.7422 - val_loss: 0.8180 - val_acc: 0.6250
Epoch 46/100
4/4 [==============================] - 1s 243ms/step - loss: 0.8160 - acc:
0.7031 - val_loss: 0.8134 - val_acc: 0.7500
Epoch 47/100
4/4 [==============================] - 1s 253ms/step - loss: 0.8118 - acc:
0.6875 - val_loss: 0.8091 - val_acc: 0.7500
Epoch 48/100
4/4 [==============================] - 1s 244ms/step - loss: 0.8071 - acc:
0.7734 - val_loss: 0.8048 - val_acc: 0.7812
Epoch 49/100
4/4 [==============================] - 1s 239ms/step - loss: 0.8029 - acc:
0.7969 - val_loss: 0.8010 - val_acc: 0.7188
Epoch 50/100
4/4 [==============================] - 1s 246ms/step - loss: 0.7983 - acc:
```

```
0.8203 - val_loss: 0.7966 - val_acc: 0.8125
Epoch 51/100
4/4 [==============================] - 1s 252ms/step - loss: 0.7943 - acc:
0.7891 - val_loss: 0.7924 - val_acc: 0.7500
Epoch 52/100
4/4 [==============================] - 1s 241ms/step - loss: 0.7898 - acc:
0.8203 - val_loss: 0.7877 - val_acc: 0.7500
Epoch 53/100
4/4 [==============================] - 1s 237ms/step - loss: 0.7844 - acc:
0.8359 - val_loss: 0.7828 - val_acc: 0.7500
Epoch 54/100
4/4 [==============================] - 1s 261ms/step - loss: 0.7784 - acc:
0.7891 - val_loss: 0.7754 - val_acc: 0.7500
Epoch 55/100
4/4 [==============================] - 1s 269ms/step - loss: 0.7679 - acc:
0.8281 - val_loss: 0.7682 - val_acc: 0.7500
Epoch 56/100
4/4 [==============================] - 1s 268ms/step - loss: 0.7588 - acc:
0.8203 - val_loss: 0.7519 - val_acc: 0.8125
Epoch 57/100
4/4 [==============================] - 1s 237ms/step - loss: 0.7368 - acc:
0.8594 - val_loss: 0.7387 - val_acc: 0.7500
Epoch 58/100
4/4 [==============================] - 1s 247ms/step - loss: 0.7095 - acc:
0.8594 - val_loss: 0.7010 - val_acc: 0.8125
Epoch 59/100
4/4 [==============================] - 1s 251ms/step - loss: 0.6659 - acc:
0.8438 - val_loss: 0.6611 - val_acc: 0.7812
Epoch 60/100
4/4 [==============================] - 1s 242ms/step - loss: 0.6195 - acc:
0.8516 - val_loss: 0.6103 - val_acc: 0.8125
Epoch 61/100
4/4 [==============================] - 1s 241ms/step - loss: 0.5748 - acc:
0.8516 - val_loss: 0.5763 - val_acc: 0.8438
Epoch 62/100
4/4 [==============================] - 1s 253ms/step - loss: 0.5040 - acc:
0.8359 - val_loss: 0.5240 - val_acc: 0.8125
Epoch 63/100
4/4 [==============================] - 1s 263ms/step - loss: 0.4171 - acc:
0.8906 - val_loss: 0.5221 - val_acc: 0.8125
Epoch 64/100
4/4 [==============================] - 1s 253ms/step - loss: 0.4229 - acc:
0.8906 - val_loss: 0.5097 - val_acc: 0.8125
Epoch 65/100
4/4 [==============================] - 1s 236ms/step - loss: 0.3183 - acc:
0.9219 - val_loss: 0.5090 - val_acc: 0.8750
Epoch 66/100
4/4 [==============================] - 1s 243ms/step - loss: 0.4026 - acc:
```

```
0.8906 - val_loss: 0.5288 - val_acc: 0.8750
Epoch 67/100
4/4 [==============================] - 1s 244ms/step - loss: 0.3819 - acc:
0.8750 - val_loss: 0.5007 - val_acc: 0.8750
Epoch 68/100
4/4 [==============================] - 1s 244ms/step - loss: 0.4305 - acc:
0.8594 - val_loss: 0.4885 - val_acc: 0.8750
Epoch 69/100
4/4 [==============================] - 1s 242ms/step - loss: 0.4459 - acc:
0.8828 - val_loss: 0.5622 - val_acc: 0.8125
Epoch 70/100
4/4 [==============================] - 1s 250ms/step - loss: 0.3266 - acc:
0.9062 - val_loss: 0.5453 - val_acc: 0.8125
Epoch 71/100
4/4 [==============================] - 1s 248ms/step - loss: 0.2843 - acc:
0.9375 - val_loss: 0.5538 - val_acc: 0.8438
Epoch 72/100
4/4 [==============================] - 1s 246ms/step - loss: 0.3527 - acc:
0.9219 - val_loss: 0.5496 - val_acc: 0.8125
Epoch 73/100
4/4 [==============================] - 1s 247ms/step - loss: 0.4462 - acc:
0.8594 - val_loss: 0.5429 - val_acc: 0.8125
Epoch 00073: early stopping
Epoch 1/100
4/4 [==============================] - 2s 516ms/step - loss: 1.9267 - acc:
0.4688 - val_loss: 1.8863 - val_acc: 0.4688
Epoch 2/100
4/4 [==============================] - 1s 229ms/step - loss: 1.8620 - acc:
0.5000 - val_loss: 1.8220 - val_acc: 0.4688
Epoch 3/100
4/4 [==============================] - 1s 242ms/step - loss: 1.7985 - acc:
0.4844 - val_loss: 1.7598 - val_acc: 0.6562
Epoch 4/100
4/4 [==============================] - 1s 243ms/step - loss: 1.7373 - acc:
0.4844 - val_loss: 1.7003 - val_acc: 0.5312
Epoch 5/100
4/4 [==============================] - 1s 248ms/step - loss: 1.6788 - acc:
0.5859 - val_loss: 1.6436 - val_acc: 0.5312
Epoch 6/100
4/4 [==============================] - 1s 248ms/step - loss: 1.6232 - acc:
0.5469 - val_loss: 1.5898 - val_acc: 0.4688
Epoch 7/100
4/4 [==============================] - 1s 261ms/step - loss: 1.5704 - acc:
0.5703 - val_loss: 1.5389 - val_acc: 0.6562
Epoch 8/100
4/4 [==============================] - 1s 249ms/step - loss: 1.5206 - acc:
0.5391 - val_loss: 1.4908 - val_acc: 0.5312
Epoch 9/100
```

```
4/4 [==============================] - 1s 247ms/step - loss: 1.4736 - acc:
0.4922 - val_loss: 1.4454 - val_acc: 0.5312
Epoch 10/100
4/4 [==============================] - 1s 240ms/step - loss: 1.4291 - acc:
0.6172 - val_loss: 1.4026 - val_acc: 0.5312
Epoch 11/100
4/4 [==============================] - 1s 244ms/step - loss: 1.3874 - acc:
0.5469 - val_loss: 1.3624 - val_acc: 0.5312
Epoch 12/100
4/4 [==============================] - 1s 256ms/step - loss: 1.3480 - acc:
0.5234 - val_loss: 1.3245 - val_acc: 0.5312
Epoch 13/100
4/4 [==============================] - 1s 265ms/step - loss: 1.3112 - acc:
0.4375 - val_loss: 1.2890 - val_acc: 0.5312
Epoch 14/100
4/4 [==============================] - 1s 245ms/step - loss: 1.2761 - acc:
0.5547 - val_loss: 1.2555 - val_acc: 0.5312
Epoch 15/100
4/4 [==============================] - 1s 242ms/step - loss: 1.2435 - acc:
0.5859 - val_loss: 1.2242 - val_acc: 0.5312
Epoch 16/100
4/4 [==============================] - 1s 243ms/step - loss: 1.2130 - acc:
0.5312 - val_loss: 1.1948 - val_acc: 0.5625
Epoch 17/100
4/4 [==============================] - 1s 244ms/step - loss: 1.1842 - acc:
0.5312 - val_loss: 1.1671 - val_acc: 0.5312
Epoch 18/100
4/4 [==============================] - 1s 246ms/step - loss: 1.1571 - acc:
0.5391 - val_loss: 1.1412 - val_acc: 0.5312
Epoch 19/100
4/4 [==============================] - 1s 249ms/step - loss: 1.1321 - acc:
0.4688 - val_loss: 1.1169 - val_acc: 0.4688
Epoch 20/100
4/4 [==============================] - 1s 245ms/step - loss: 1.1083 - acc:
0.4922 - val_loss: 1.0941 - val_acc: 0.4688
Epoch 21/100
4/4 [==============================] - 1s 244ms/step - loss: 1.0859 - acc:
0.5391 - val_loss: 1.0727 - val_acc: 0.5312
Epoch 22/100
4/4 [==============================] - 1s 242ms/step - loss: 1.0650 - acc:
0.5000 - val_loss: 1.0526 - val_acc: 0.5312
Epoch 23/100
4/4 [==============================] - 1s 244ms/step - loss: 1.0452 - acc:
0.5625 - val_loss: 1.0338 - val_acc: 0.7500
Epoch 24/100
4/4 [==============================] - 1s 247ms/step - loss: 1.0271 - acc:
0.5547 - val_loss: 1.0162 - val_acc: 0.7812
Epoch 25/100
```

```
4/4 [==============================] - 1s 245ms/step - loss: 1.0100 - acc:
0.4844 - val_loss: 0.9997 - val_acc: 0.4688
Epoch 26/100
4/4 [==============================] - 1s 242ms/step - loss: 0.9939 - acc:
0.4844 - val_loss: 0.9842 - val_acc: 0.4688
Epoch 27/100
4/4 [==============================] - 1s 241ms/step - loss: 0.9786 - acc:
0.5156 - val_loss: 0.9696 - val_acc: 0.4688
Epoch 28/100
4/4 [==============================] - 1s 256ms/step - loss: 0.9642 - acc:
0.5391 - val_loss: 0.9560 - val_acc: 0.4688
Epoch 29/100
4/4 [==============================] - 1s 243ms/step - loss: 0.9513 - acc:
0.4766 - val_loss: 0.9432 - val_acc: 0.4688
Epoch 30/100
4/4 [==============================] - 1s 240ms/step - loss: 0.9387 - acc:
0.4922 - val_loss: 0.9312 - val_acc: 0.4688
Epoch 31/100
4/4 [==============================] - 1s 237ms/step - loss: 0.9269 - acc:
0.4609 - val_loss: 0.9199 - val_acc: 0.4688
Epoch 32/100
4/4 [==============================] - 1s 253ms/step - loss: 0.9159 - acc:
0.4453 - val_loss: 0.9093 - val_acc: 0.4688
Epoch 33/100
4/4 [==============================] - 1s 248ms/step - loss: 0.9056 - acc:
0.5391 - val_loss: 0.8993 - val_acc: 0.4688
Epoch 34/100
4/4 [==============================] - 1s 268ms/step - loss: 0.8958 - acc:
0.4766 - val_loss: 0.8900 - val_acc: 0.5625
Epoch 35/100
4/4 [==============================] - 1s 241ms/step - loss: 0.8867 - acc:
0.4922 - val_loss: 0.8812 - val_acc: 0.5312
Epoch 36/100
4/4 [==============================] - 1s 244ms/step - loss: 0.8781 - acc:
0.5234 - val_loss: 0.8730 - val_acc: 0.6562
Epoch 37/100
4/4 [==============================] - 1s 243ms/step - loss: 0.8701 - acc:
0.4766 - val_loss: 0.8653 - val_acc: 0.6562
Epoch 38/100
4/4 [==============================] - 1s 248ms/step - loss: 0.8625 - acc:
0.5078 - val_loss: 0.8580 - val_acc: 0.5312
Epoch 39/100
4/4 [==============================] - 1s 243ms/step - loss: 0.8554 - acc:
0.5391 - val_loss: 0.8512 - val_acc: 0.5312
Epoch 40/100
4/4 [==============================] - 1s 243ms/step - loss: 0.8487 - acc:
0.5391 - val_loss: 0.8448 - val_acc: 0.5312
Epoch 41/100
```

```
4/4 [==============================] - 1s 250ms/step - loss: 0.8425 - acc:
0.4844 - val_loss: 0.8387 - val_acc: 0.5312
Epoch 42/100
4/4 [==============================] - 1s 245ms/step - loss: 0.8366 - acc:
0.5156 - val_loss: 0.8330 - val_acc: 0.5312
Epoch 43/100
4/4 [==============================] - 1s 240ms/step - loss: 0.8310 - acc:
0.5469 - val_loss: 0.8277 - val_acc: 0.5312
Epoch 44/100
4/4 [==============================] - 1s 244ms/step - loss: 0.8259 - acc:
0.4453 - val_loss: 0.8226 - val_acc: 0.5312
Epoch 45/100
4/4 [==============================] - 1s 244ms/step - loss: 0.8209 - acc:
0.4219 - val_loss: 0.8179 - val_acc: 0.5625
Epoch 46/100
4/4 [==============================] - 1s 247ms/step - loss: 0.8162 - acc:
0.4531 - val_loss: 0.8134 - val_acc: 0.5312
Epoch 47/100
4/4 [==============================] - 1s 243ms/step - loss: 0.8118 - acc:
0.5000 - val_loss: 0.8091 - val_acc: 0.5312
Epoch 48/100
4/4 [==============================] - 1s 241ms/step - loss: 0.8077 - acc:
0.4766 - val_loss: 0.8051 - val_acc: 0.4688
Epoch 49/100
4/4 [==============================] - 1s 248ms/step - loss: 0.8038 - acc:
0.4766 - val_loss: 0.8013 - val_acc: 0.5000
Epoch 50/100
4/4 [==============================] - 1s 248ms/step - loss: 0.7999 - acc:
0.4922 - val_loss: 0.7977 - val_acc: 0.4688
Epoch 51/100
4/4 [==============================] - 1s 237ms/step - loss: 0.7963 - acc:
0.5156 - val_loss: 0.7943 - val_acc: 0.4688
Epoch 52/100
4/4 [==============================] - 1s 252ms/step - loss: 0.7929 - acc:
0.5078 - val_loss: 0.7910 - val_acc: 0.4688
Epoch 53/100
4/4 [==============================] - 1s 261ms/step - loss: 0.7898 - acc:
0.5000 - val_loss: 0.7879 - val_acc: 0.4688
Epoch 54/100
4/4 [==============================] - 1s 257ms/step - loss: 0.7867 - acc:
0.5234 - val_loss: 0.7850 - val_acc: 0.4688
Epoch 55/100
4/4 [==============================] - 1s 263ms/step - loss: 0.7839 - acc:
0.5391 - val_loss: 0.7822 - val_acc: 0.4688
Epoch 56/100
4/4 [==============================] - 1s 251ms/step - loss: 0.7811 - acc:
0.4922 - val_loss: 0.7795 - val_acc: 0.4688
Epoch 57/100
```

```
4/4 [==============================] - 1s 250ms/step - loss: 0.7785 - acc:
0.5078 - val_loss: 0.7770 - val_acc: 0.4688
Epoch 58/100
4/4 [==============================] - 1s 245ms/step - loss: 0.7760 - acc:
0.5000 - val_loss: 0.7746 - val_acc: 0.4688
Epoch 59/100
4/4 [==============================] - 1s 245ms/step - loss: 0.7736 - acc:
0.5312 - val_loss: 0.7722 - val_acc: 0.4688
Epoch 60/100
4/4 [==============================] - 1s 241ms/step - loss: 0.7714 - acc:
0.4766 - val_loss: 0.7700 - val_acc: 0.4688
Epoch 61/100
4/4 [==============================] - 1s 245ms/step - loss: 0.7691 - acc:
0.5000 - val_loss: 0.7679 - val_acc: 0.4688
Epoch 62/100
4/4 [==============================] - 1s 245ms/step - loss: 0.7670 - acc:
0.5000 - val_loss: 0.7658 - val_acc: 0.4688
Epoch 63/100
4/4 [==============================] - 1s 242ms/step - loss: 0.7651 - acc:
0.4844 - val_loss: 0.7639 - val_acc: 0.4688
Epoch 64/100
4/4 [==============================] - 1s 245ms/step - loss: 0.7631 - acc:
0.4922 - val_loss: 0.7620 - val_acc: 0.4688
Epoch 65/100
4/4 [==============================] - 1s 244ms/step - loss: 0.7613 - acc:
0.4922 - val_loss: 0.7602 - val_acc: 0.4688
Epoch 66/100
4/4 [==============================] - 1s 252ms/step - loss: 0.7595 - acc:
0.5078 - val_loss: 0.7584 - val_acc: 0.4688
Epoch 67/100
4/4 [==============================] - 1s 265ms/step - loss: 0.7578 - acc:
0.4844 - val_loss: 0.7567 - val_acc: 0.4688
Epoch 68/100
4/4 [==============================] - 1s 246ms/step - loss: 0.7561 - acc:
0.5625 - val_loss: 0.7551 - val_acc: 0.4688
Epoch 69/100
4/4 [==============================] - 1s 245ms/step - loss: 0.7545 - acc:
0.4688 - val_loss: 0.7535 - val_acc: 0.4688
Epoch 70/100
4/4 [==============================] - 1s 247ms/step - loss: 0.7529 - acc:
0.4922 - val_loss: 0.7520 - val_acc: 0.4688
Epoch 71/100
4/4 [==============================] - 1s 249ms/step - loss: 0.7514 - acc:
0.4531 - val_loss: 0.7505 - val_acc: 0.4688
Epoch 72/100
4/4 [==============================] - 1s 242ms/step - loss: 0.7501 - acc:
0.4219 - val_loss: 0.7491 - val_acc: 0.4688
Epoch 73/100
```

```
4/4 [==============================] - 1s 253ms/step - loss: 0.7486 - acc:
0.4844 - val_loss: 0.7477 - val_acc: 0.4688
Epoch 74/100
4/4 [==============================] - 1s 245ms/step - loss: 0.7471 - acc:
0.5469 - val_loss: 0.7464 - val_acc: 0.4688
Epoch 75/100
4/4 [==============================] - 1s 252ms/step - loss: 0.7458 - acc:
0.5078 - val_loss: 0.7451 - val_acc: 0.4688
Epoch 76/100
4/4 [==============================] - 1s 265ms/step - loss: 0.7447 - acc:
0.4688 - val_loss: 0.7439 - val_acc: 0.4688
Epoch 77/100
4/4 [==============================] - 1s 241ms/step - loss: 0.7434 - acc:
0.5234 - val_loss: 0.7426 - val_acc: 0.4688
Epoch 78/100
4/4 [==============================] - 1s 247ms/step - loss: 0.7421 - acc:
0.5391 - val_loss: 0.7415 - val_acc: 0.4688
Epoch 79/100
4/4 [==============================] - 1s 245ms/step - loss: 0.7410 - acc:
0.5156 - val_loss: 0.7403 - val_acc: 0.4688
Epoch 80/100
4/4 [==============================] - 1s 241ms/step - loss: 0.7398 - acc:
0.5234 - val_loss: 0.7392 - val_acc: 0.4688
Epoch 81/100
4/4 [==============================] - 1s 244ms/step - loss: 0.7389 - acc:
0.4375 - val_loss: 0.7381 - val_acc: 0.4688
Epoch 82/100
4/4 [==============================] - 1s 244ms/step - loss: 0.7377 - acc:
0.5156 - val_loss: 0.7370 - val_acc: 0.4688
Epoch 83/100
4/4 [==============================] - 1s 244ms/step - loss: 0.7366 - acc:
0.5469 - val_loss: 0.7360 - val_acc: 0.4688
Epoch 84/100
4/4 [==============================] - 1s 269ms/step - loss: 0.7356 - acc:
0.5156 - val_loss: 0.7350 - val_acc: 0.4688
Epoch 85/100
4/4 [==============================] - 1s 244ms/step - loss: 0.7347 - acc:
0.4531 - val_loss: 0.7340 - val_acc: 0.4688
Epoch 86/100
4/4 [==============================] - 1s 249ms/step - loss: 0.7337 - acc:
0.4766 - val_loss: 0.7331 - val_acc: 0.4688
Epoch 87/100
4/4 [==============================] - 1s 244ms/step - loss: 0.7327 - acc:
0.5000 - val_loss: 0.7321 - val_acc: 0.4688
Epoch 88/100
4/4 [==============================] - 1s 237ms/step - loss: 0.7317 - acc:
0.6094 - val_loss: 0.7312 - val_acc: 0.4688
Epoch 89/100
```

```
4/4 [==============================] - 1s 256ms/step - loss: 0.7309 - acc:
0.5078 - val_loss: 0.7303 - val_acc: 0.4688
Epoch 90/100
4/4 [==============================] - 1s 242ms/step - loss: 0.7300 - acc:
0.5391 - val_loss: 0.7295 - val_acc: 0.4688
Epoch 91/100
4/4 [==============================] - 1s 245ms/step - loss: 0.7292 - acc:
0.4688 - val_loss: 0.7286 - val_acc: 0.4688
Epoch 92/100
4/4 [==============================] - 1s 244ms/step - loss: 0.7283 - acc:
0.5312 - val_loss: 0.7278 - val_acc: 0.4688
Epoch 93/100
4/4 [==============================] - 1s 246ms/step - loss: 0.7276 - acc:
0.5078 - val_loss: 0.7270 - val_acc: 0.4688
Epoch 94/100
4/4 [==============================] - 1s 244ms/step - loss: 0.7267 - acc:
0.5156 - val_loss: 0.7262 - val_acc: 0.4688
Epoch 95/100
4/4 [==============================] - 1s 242ms/step - loss: 0.7260 - acc:
0.5156 - val_loss: 0.7255 - val_acc: 0.4688
Epoch 96/100
4/4 [==============================] - 1s 249ms/step - loss: 0.7251 - acc:
0.5469 - val_loss: 0.7247 - val_acc: 0.4688
Epoch 97/100
4/4 [==============================] - 1s 242ms/step - loss: 0.7244 - acc:
0.4453 - val_loss: 0.7240 - val_acc: 0.4688
Epoch 98/100
4/4 [==============================] - 1s 242ms/step - loss: 0.7237 - acc:
0.5234 - val_loss: 0.7233 - val_acc: 0.4688
Epoch 99/100
4/4 [==============================] - 1s 250ms/step - loss: 0.7230 - acc:
0.5234 - val_loss: 0.7226 - val_acc: 0.4688
Epoch 100/100
4/4 [==============================] - 1s 253ms/step - loss: 0.7223 - acc:
0.4922 - val_loss: 0.7219 - val_acc: 0.4688
Individual model 1 - Loss: 45.87810516357422, Accuracy: 0.550000011920929
Individual model 2 - Loss: 0.7218674898147583, Accuracy: 0.5249999761581421
Ensemble model - Loss: 0.09719023555517196, Accuracy: 1.0
```

[ ]:
```python
# Evaluate individual models
loss1, acc1 = model_cnn3.evaluate(test_data, test_labels, verbose=0)
loss2, acc2 = model_cnn4.evaluate(test_data, test_labels, verbose=0)

# Evaluate ensemble model
ensemble_loss, ensemble_acc = model_cnn3.evaluate(test_data,
    ensemble_pred_binary, verbose=0)
```

```
# display results
print('Individual model 1 - Loss: {}, Accuracy: {}'.format(loss1, acc1))
print('Individual model 2 - Loss: {}, Accuracy: {}'.format(loss2, acc2))
print('Ensemble model - Loss: {}, Accuracy: {}'.format(ensemble_loss,␣
↪ensemble_acc))
```

```
Individual model 1 - Loss: 45.87810516357422, Accuracy: 0.550000011920929
Individual model 2 - Loss: 0.7218674898147583, Accuracy: 0.5249999761581421
Ensemble model - Loss: 0.09719023555517196, Accuracy: 1.0
```

The network architecture used in the model is a CNN (Convolutional Neural Network) consisting of four convolutional layers, each followed by a max-pooling layer, and two fully connected (dense) layers. The first convolutional layer has 32 filters of size 3x3 with a ReLU activation function. The second convolutional layer has 64 filters of size 3x3 with a ReLU activation function. The third convolutional layer has 128 filters of size 3x3 with a ReLU activation function. The fourth convolutional layer also has 128 filters of size 3x3 with a ReLU activation function. Each max-pooling layer has a pool size of 2x2. The first dense layer has 512 units with a ReLU activation function, and the second dense layer has a single output unit with a sigmoid activation function. The model uses the Adam optimizer with a learning rate of 0.0001, binary cross-entropy loss function, and accuracy as the evaluation metric.

To monitor the convergence of the model, the EarlyStopping callback was used. EarlyStopping stops the training process if there is no improvement in the validation loss over a certain number of epochs (patience). In this model, the EarlyStopping callback was set to monitor the validation loss, with a patience of 5 epochs. If the validation loss did not improve for 5 epochs, the training process would stop.

To prevent overfitting, several techniques were used. First, data augmentation was performed using the ImageDataGenerator function, which randomly applies transformations such as rotation, zooming, shifting, and flipping to the training images. This technique increases the number of training samples and helps the model generalize better. Second, dropout regularization was applied to the first dense layer with a rate of 0.5. Dropout randomly sets a fraction of the inputs to zero during training, which reduces overfitting by forcing the model to learn more robust features. Third, L2 regularization was applied to the convolutional and dense layers with a weight of 0.001. L2 regularization adds a penalty term to the loss function that discourages large weight values, which helps prevent overfitting by reducing the complexity of the model. Finally, the EarlyStopping callback was used as mentioned earlier to stop the training process if the validation loss did not improve for a certain number of epochs.

[ ]:
```
# define the FNN model
model_fnn = Sequential()
model_fnn.add(Dense(512, activation='relu', input_shape=(150528,)))
model_fnn.add(Dropout(0.5))
model_fnn.add(Dense(256, activation='relu'))
model_fnn.add(Dropout(0.5))
model_fnn.add(Dense(1, activation='sigmoid'))

# compile the FNN model
```

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

# fit the FNN model
history_fnn = model_fnn.fit(train_data.reshape((train_data.shape[0], -1)),␣
 ↪train_labels, epochs=50, batch_size=32, validation_data=(val_data.
 ↪reshape((val_data.shape[0], -1)), val_labels), callbacks=[es])
```

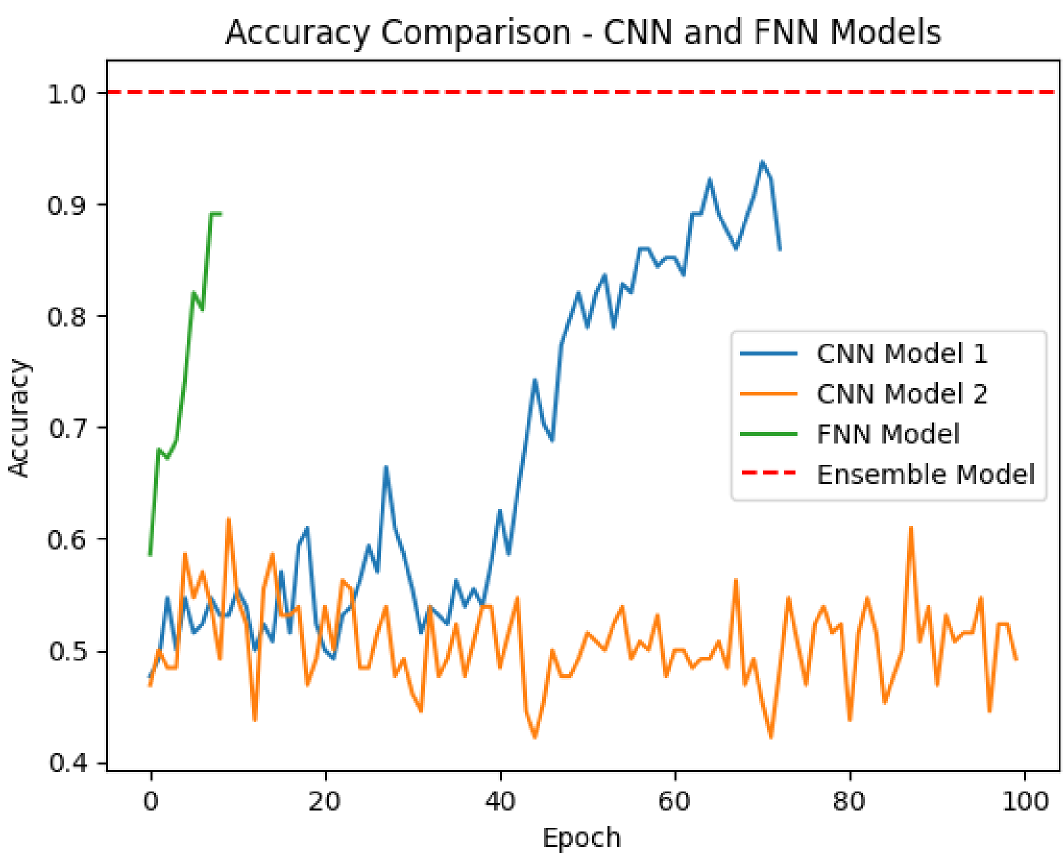
```
Train on 128 samples, validate on 32 samples
Epoch 1/50
128/128 [==============================] - 0s 3ms/sample - loss: 31.2605 - acc:
0.5859 - val_loss: 35.9631 - val_acc: 0.5312
Epoch 2/50
128/128 [==============================] - 0s 1ms/sample - loss: 44.0968 - acc:
0.6797 - val_loss: 26.6617 - val_acc: 0.5312
Epoch 3/50
128/128 [==============================] - 0s 1ms/sample - loss: 36.3502 - acc:
0.6719 - val_loss: 55.7036 - val_acc: 0.5625
Epoch 4/50
128/128 [==============================] - 0s 1ms/sample - loss: 42.6521 - acc:
0.6875 - val_loss: 10.2732 - val_acc: 0.8438
Epoch 5/50
128/128 [==============================] - 0s 1ms/sample - loss: 36.0089 - acc:
0.7422 - val_loss: 22.4459 - val_acc: 0.8125
Epoch 6/50
128/128 [==============================] - 0s 1ms/sample - loss: 13.7583 - acc:
0.8203 - val_loss: 55.0828 - val_acc: 0.7500
Epoch 7/50
128/128 [==============================] - 0s 1ms/sample - loss: 37.4660 - acc:
0.8047 - val_loss: 29.3118 - val_acc: 0.7500
Epoch 8/50
128/128 [==============================] - 0s 1ms/sample - loss: 10.9626 - acc:
0.8906 - val_loss: 16.6657 - val_acc: 0.8438
Epoch 9/50
128/128 [==============================] - 0s 1ms/sample - loss: 16.9209 - acc:
0.8906 - val_loss: 14.8229 - val_acc: 0.9062
Epoch 00009: early stopping
```

[ ]:
```
# evaluate CNN model
# test_loss_cnn, test_acc_cnn = model_cnn.evaluate(test_data, test_labels)

# evaluate FNN model
test_loss_fnn, test_acc_fnn = model_fnn.evaluate(test_data.reshape((test_data.
 ↪shape[0], -1)), test_labels)

# print('CNN model accuracy:', test_acc_cnn)
print('FNN model accuracy:', test_acc_fnn)
```

```
print('FNN model loss:', test_loss_fnn)
```

```
40/40 [==============================] - 0s 2ms/sample - loss: 13.4833 - acc:
0.8250
FNN model accuracy: 0.825
FNN model loss: 13.483303260803222
```

[ ]:
```
import matplotlib.pyplot as plt

# Plot individual model accuracies
plt.plot(history1.history['acc'], label='CNN Model 1')
plt.plot(history2.history['acc'], label='CNN Model 2')
plt.plot(history_fnn.history['acc'], label='FNN Model')

# Plot ensemble model accuracy
plt.axhline(y=ensemble_acc, color='r', linestyle='--', label='Ensemble Model')

plt.title('Accuracy Comparison - CNN and FNN Models')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



2 Why CNN is better in image classification than FNN

When it comes to image classification, Convolutional Neural Networks (CNNs) are generally considered more suitable than Fully Connected Neural Networks (FNNs). This is because CNNs are designed to take advantage of the spatial structure of images, which is something FNNs are not equipped to do.

In an FNN, every neuron in each layer is connected to every neuron in the previous layer. This means that as the number of neurons in each layer increases, the number of weights and connections in the network also grows exponentially. This can make FNNs prohibitively expensive to train and computationally expensive to use for large image datasets.

In contrast, CNNs are designed to exploit the spatial locality and correlation present in images. They use convolutional layers to scan an image with a set of learnable filters, extracting important features while ignoring irrelevant information. This allows them to capture the hierarchical structure of an image, from simple edges and shapes to more complex features like eyes and noses, using fewer parameters and computations than an FNN.

Additionally, CNNs often incorporate pooling layers, which downsample the output of the convolutional layers by taking the maximum or average value within a small window. This helps reduce the dimensionality of the data and provide some degree of translation invariance, meaning that the CNN can still recognize an object even if it appears in a slightly different position or orientation within the image.

Overall, CNNs are better suited for image classification tasks than FNNs because they can capture the spatial structure of images using fewer parameters and computations. This makes them faster, more efficient, and more effective at recognizing patterns and features within large datasets of images.

3 Subtask 2 : Using pretrained ResNet-50 model

[37]:
```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras import regularizers

# load the pre-trained ResNet-50 model
resnet = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224,␣
↪3))

# freeze the weights of the pre-trained model
for layer in resnet.layers:
```

```python
    layer.trainable = False

# add new trainable layers
x = resnet.output
x = Flatten()(x)
x = Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.01))(x)
x = Dropout(0.5)(x)
x = Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.01))(x)
x = Dropout(0.5)(x)
output = Dense(2, activation='softmax')(x)

# create the new model
model = Model(inputs=resnet.input, outputs=output)

# learning rate
opt = Adam(lr=0.0001)

# compile the model
model.compile(optimizer=opt, loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# create data generators for training and validation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True,
    width_shift_range=0.2,
    height_shift_range=0.2,
    validation_split=0.2)
train_generator = train_datagen.flow(train_data, train_labels, batch_size=32,␣
 ↪subset='training')
val_generator = train_datagen.flow(train_data, train_labels, batch_size=32,␣
 ↪subset='validation')

# set up early stopping callback
es = EarlyStopping(monitor='val_loss', mode='min', patience=5)

# train the model
history = model.fit(train_generator, epochs=100, validation_data=val_generator,␣
 ↪callbacks=[es], batch_size=32)

# history = model.fit(train_data, train_labels, epochs=10,␣
 ↪validation_data=(val_data, val_labels))
```

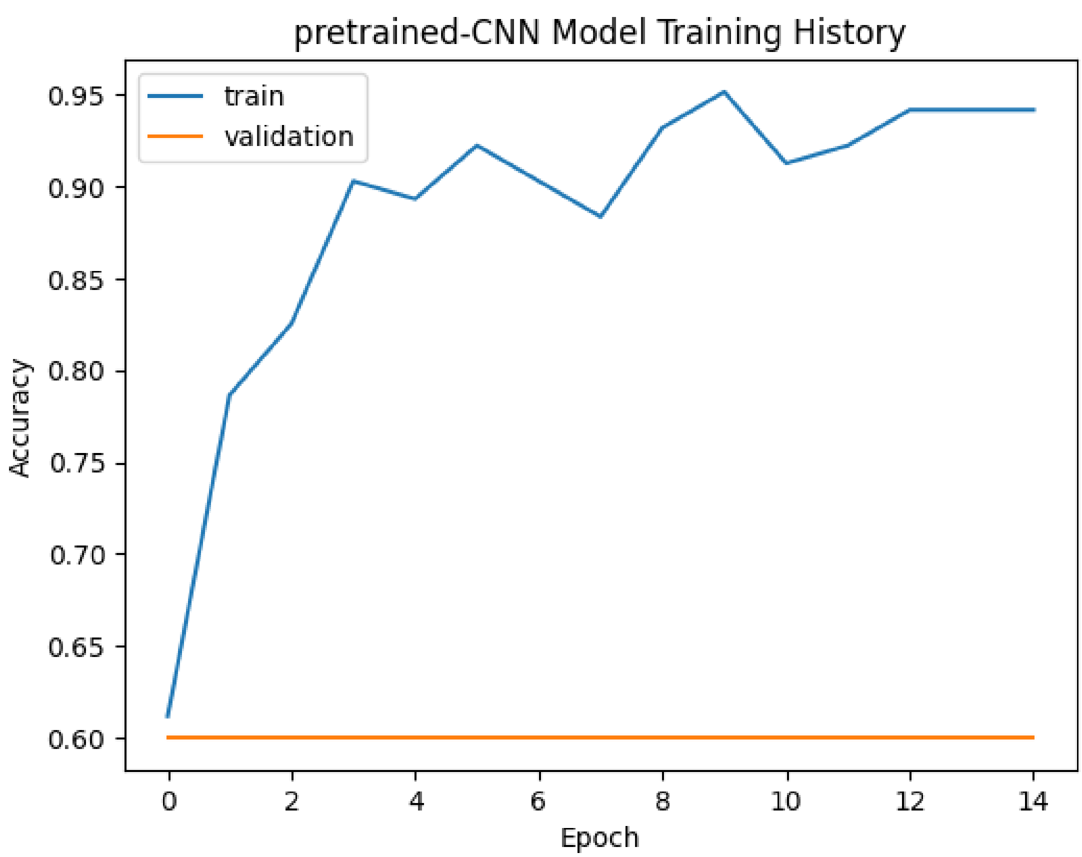
```
c:\Users\abhij\.conda\envs\image_ML\lib\site-
packages\keras_applications\resnet50.py:265: UserWarning: The output shape of
```

```
`ResNet50(include_top=False)` has been changed since Keras 2.2.0.
  warnings.warn('The output shape of `ResNet50(include_top=False)` '

Epoch 1/100
4/4 [==============================] - 6s 1s/step - loss: 8.2409 - acc: 0.6117 -
val_loss: 7.2912 - val_acc: 0.6000
Epoch 2/100
4/4 [==============================] - 1s 185ms/step - loss: 7.4592 - acc:
0.7864 - val_loss: 7.4052 - val_acc: 0.6000
Epoch 3/100
4/4 [==============================] - 1s 247ms/step - loss: 7.5789 - acc:
0.8252 - val_loss: 7.3059 - val_acc: 0.6000
Epoch 4/100
4/4 [==============================] - 1s 246ms/step - loss: 6.8934 - acc:
0.9029 - val_loss: 7.1274 - val_acc: 0.6000
Epoch 5/100
4/4 [==============================] - 1s 217ms/step - loss: 7.0916 - acc:
0.8932 - val_loss: 7.1916 - val_acc: 0.6000
Epoch 6/100
4/4 [==============================] - 1s 214ms/step - loss: 6.7520 - acc:
0.9223 - val_loss: 7.3140 - val_acc: 0.6000
Epoch 7/100
4/4 [==============================] - 1s 254ms/step - loss: 6.8952 - acc:
0.9029 - val_loss: 7.2473 - val_acc: 0.6000
Epoch 8/100
4/4 [==============================] - 1s 226ms/step - loss: 6.5618 - acc:
0.8835 - val_loss: 7.0879 - val_acc: 0.6000
Epoch 9/100
4/4 [==============================] - 1s 242ms/step - loss: 6.4489 - acc:
0.9320 - val_loss: 6.8326 - val_acc: 0.6000
Epoch 10/100
4/4 [==============================] - 1s 248ms/step - loss: 6.1875 - acc:
0.9515 - val_loss: 6.6866 - val_acc: 0.6000
Epoch 11/100
4/4 [==============================] - 1s 236ms/step - loss: 6.2168 - acc:
0.9126 - val_loss: 6.7202 - val_acc: 0.6000
Epoch 12/100
4/4 [==============================] - 1s 210ms/step - loss: 6.2019 - acc:
0.9223 - val_loss: 6.7054 - val_acc: 0.6000
Epoch 13/100
4/4 [==============================] - 1s 224ms/step - loss: 6.2116 - acc:
0.9417 - val_loss: 6.7371 - val_acc: 0.6000
Epoch 14/100
4/4 [==============================] - 1s 205ms/step - loss: 6.0112 - acc:
0.9417 - val_loss: 6.9264 - val_acc: 0.6000
Epoch 15/100
4/4 [==============================] - 1s 229ms/step - loss: 6.0264 - acc:
0.9417 - val_loss: 6.9319 - val_acc: 0.6000
```

[38]:
```
# evaluate the model
_, train_acc = model.evaluate(train_data, train_labels, verbose=0)
_, val_acc = model.evaluate(val_data, val_labels, verbose=0)
_, test_acc = model.evaluate(test_data, test_labels, verbose=0)
print('Training accuracy: %.3f, Validation accuracy: %.3f, Test accuracy: %.3f'␣
↪% (train_acc, val_acc, test_acc))
```

```
Training accuracy: 0.500, Validation accuracy: 0.469, Test accuracy: 0.525
```

[39]:
```
import matplotlib.pyplot as plt

# plot pretrained-CNN training history
plt.plot(history.history['acc'], label='train')
plt.plot(history.history['val_acc'], label='validation')
plt.title('pretrained-CNN Model Training History')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



ResNet50 is a deep neural network with 50 layers that was trained on the ImageNet dataset, which contains over 14 million labeled images. It has been shown to perform very well on a wide range of image recognition tasks. However, when I used it to classify with the current images, you noticed that it was overfitting the training data and not generalizing well to the validation and test data.

Overfitting occurs when a model learns the training data too well and becomes too specialized to that particular dataset. This can happen when the model is too complex relative to the size of the training data or when the model is trained for too many epochs.

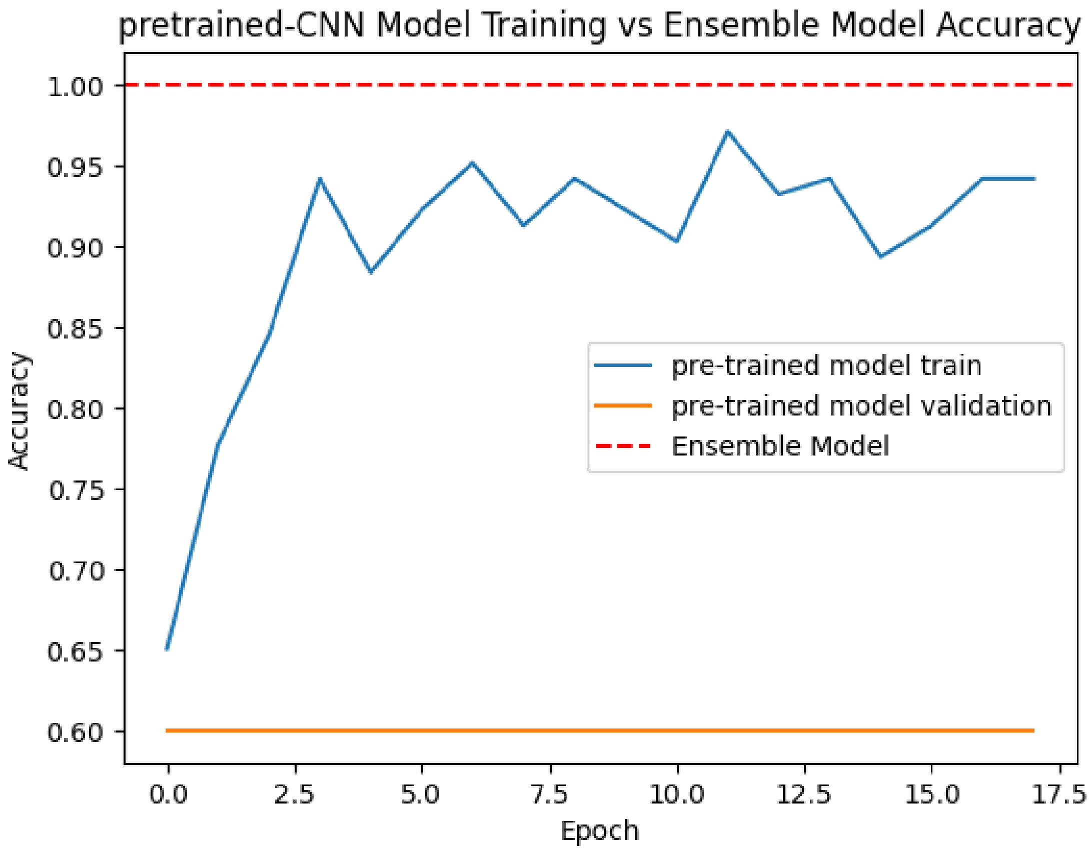
To prevent overfitting, I tried several strategies. One approach was to use regularization techniques such as L1 or L2 regularization, dropout, or data augmentation. Another approach was to reduce the complexity of the model by removing some layers or reducing the number of nodes in each layer.

In this case, the CNN model built from scratch performed better than the pre-trained ResNet50 model because the former was specifically designed and trained to this specific project, whereas the latter was trained on a different dataset and may not have been optimized for your specific task.

To perform transfer learning with ResNet50, pre-trained model was loaded and removed the final classification layer. Added a new classification layer and froze all the layers in the pre-trained model except for the new classification layer. This allowed me to use the pre-trained weights as a starting point for my own training process and fine-tune the model to this specific task.

However, in this case, I may have encountered overfitting when fine-tuning the model because the new classification layer was not very large and the dataset was relatively small. Freezing some of the earlier layers may have helped prevent overfitting because it would have prevented the model from learning too much from the small dataset and relying too heavily on the pre-trained weights.

[ ]:
```
# Compare the performance of the pretrained-CNN model with the CNN_ensemble␣
↪model

import matplotlib.pyplot as plt

plt.plot(history.history['acc'], label='pre-trained model train')
plt.plot(history.history['val_acc'], label='pre-trained model validation')
# Plot ensemble model accuracy
plt.axhline(y=ensemble_acc, color='r', linestyle='--', label='Ensemble Model')

plt.title('pretrained-CNN Model Training vs Ensemble Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



pretrained-CNN Model Training vs Ensemble Model Accuracy
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
pre-trained model train
pre-trained model validation
Ensemble Model
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch