Introduction to AI

Assignment 1

February 22, 2023

1. [1] Write True/False for the following conditional independence statements. Justify clearly your answer by showing active/blocked trails as necessary and appropriate rules for them to be active/blocked. [No coding required for this question. Each sub-question has **2 points**]

   (a) $A \perp G \mid \{F\}$
   False. There is a V-structure between A and G.

   $$A \rightarrow B \rightarrow D \leftarrow G$$

   Given F and thus implying we know D, it couples A and G. Thus A and G are dependent.

   (b) $A \perp G \mid \{E\}$
   False. There is a common cause structure.

   $$A \leftarrow C \rightarrow E \rightarrow G$$

   A and E are dependent, and thus by cascade structure A and G are also dependent. But given E, it blocks the active path between A and G. Thus A and G are independent given E.

   (c) $A \perp E \mid \{C\}$
   True. A, C and E have a common cause structure.

   $$A \leftarrow C \rightarrow E$$

   Given the common cause C, it decouples A and E. There are no other active paths between A and C, thus they are independent.

   (d) $A \perp E \mid \{C, D\}$
   False. Similar to part (c), given C it decouples A and E at $A \leftarrow C \rightarrow E$. However, there exists a V-structure between A and E.

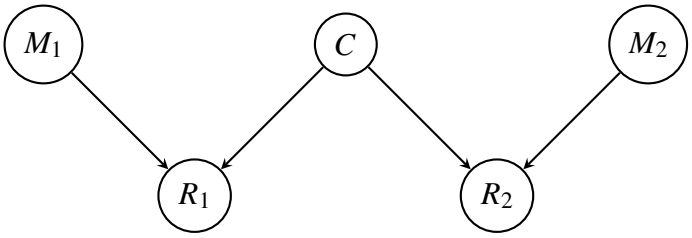
   $$A \rightarrow B \rightarrow D \leftarrow E$$

   Similar to part (a), given D it couples A and E. Thus A and E are dependent.

   (e) $A \perp D \mid \{B, E\}$
   True. Cascade from $A \rightarrow B \rightarrow D$ is blocked given B. Common cause couples A and E. But path from $A \leftarrow C \rightarrow E \rightarrow D$ is blocked given E. Since all paths from A to D are blocked, A and D are independent.

[1] Full submission can be found on https://github.com/jinhanloh2021/AI_AS1

2. (a) Draw the Bayes net corresponding to this setup. **[3 points]**



Variable Name	Domain	Interpretation
C	$\{1, 0\}$	The actual health of a person. Either COVID positive 1 or negative 0.
M_1	$\{a, b, c\}$	The manufacturer of the first test kit. Where the company is a, b or c.
M_2	$\{a, b, c\}$	The manufacturer of the second test kit. Where the company is a, b or c.
R_1	$\{1, 0\}$	The result of the first test kit. Either positive 1 or negative 0.
R_2	$\{1, 0\}$	The result of the second test kit. Either positive 1 or negative 0.

This table can be interpreted as three independent events M_1, M_2 and C. The manufacturer of the two test kits received by a person and their actual health are independent, but they will determine the result of the test kit.

(b) Write conditional probabilities (numerical values) associated with each node of this Bayes net. As there are 5 variables, please specify one conditional probability table (CPT) for each variable. **[2 points]**

$P(C = 0)$	$P(C = 1)$
0.7	0.3

$P(M_n = a)$	$P(M_n = b)$	$P(M_n = c)$
0.333	0.333	0.333

M_n	C	$P(R_n = 0 \mid M_n, C)$	$P(R_n = 1 \mid M_n, C)$
a	0	0.99	0.01
b	0	0.95	0.05
c	0	0.91	0.09
a	1	0.3	0.7
b	1	0.2	0.8
c	1	0.1	0.9

For values of $n \in \{1, 2\}$ as each person has two test kits.

(c) Are the results of the two tests dependent or independent given the evidence that the Covid Status is known? Justify your answer. **[1 point]**

There is a common cause structure $R_1 \leftarrow C \rightarrow R_2$ which couples R_1 and R_2. But given the COVID status C, it decouples R_1 and R_2. Thus they are independent.

(d) Assume you took both tests at home. After being tested twice in a matter of minutes, the first test was positive and the second negative. What is the probability that you actually have COVID19? Show your analytical computations. **[4 points]**

Given that the first test was positive and the second test was negative, then

$$\begin{aligned} R_1 &= 1 \\ R_2 &= 0 \end{aligned}$$

We are trying to solve for $P(C = 1 \mid R_1 = 1, R_2 = 0)$. Since R_1 and R_2 are conditionally independent on C,

$$\begin{aligned} P(C = 1 \mid R_1 = 1, R_2 = 0) &= \frac{P(R_1 = 1, R_2 = 0 \mid C = 1)P(C = 1)}{P(R_1 = 1, R_2 = 0)} \\ &= \frac{P(R_1 = 1 \mid C = 1)P(R_2 = 0 \mid C = 1)P(C = 1)}{P(R_1 = 1, R_2 = 0)} \end{aligned}$$

To find $P(R_1 = 1 \mid C = 1)$ and $P(R_2 = 0 \mid C = 1)$, marginalise over the manufacturer M.

$$\begin{aligned} P(R_1 = 1 \mid C = 1) &= \frac{0.7 + 0.8 + 0.9}{(0.1 + 0.2 + 0.3) + (0.7 + 0.8 + 0.9)} \\ &= 0.8 \\ P(R_2 = 0 \mid C = 1) &= 1 - 0.8 \\ &= 0.2 \end{aligned}$$

Thus we have solved for numerator,

$$\begin{aligned} P(R_1 = 1 \mid C = 1)P(R_2 = 0 \mid C = 1)P(C = 1) &= (0.8)(0.2)(0.3) \\ &= 0.048 \end{aligned}$$

To solve for the denominator, it is the *hard part*. We cannot use marginalisation as there are too many varibales to sum over. We can use variable elimination instead. The joint distribution is

$$\begin{aligned} &P(C = c, M_1 = m_1, M_2 = m_2, R_1 = r_1, R_2 = r_2) \\ =&P(C = c)P(M_1 = m_1)P(M_2 = m_2)P(R_1 = r_1 \mid C, M_1)P(R_2 = r_2 \mid C, M_2) \end{aligned}$$

Then,

$$
\begin{aligned}
& P(R_1 = 1, R_2 = 0) \\
&= \sum_{C,M_1,M_2} P(C = c)P(M_1 = m_1)P(M_2 = m_2)P(R_1 = 1 \mid C, M_1)P(R_2 = 0 \mid C, M_2) \\
&= \sum_{C,M_1} P(C = c)P(M_1 = m_1)P(R_1 = 1 \mid C, M_1) \underbrace{\sum_{M_2} P(M_2 = m_2)P(R_2 = 0 \mid C, M_2)}_{\tau_1(R_2=0|C)} \\
&= \sum_{C} P(C = c)\tau_1(R_2 = 0 \mid C) \underbrace{\sum_{M_1} (M_1 = m_1)P(R_1 = 1 \mid C, M_1)}_{\tau_2(R_1=1|C)} \\
&= \sum_{C} P(C = c)\tau_1(R_2 = 0 \mid C)\tau_2(R_1 = 1 \mid C)
\end{aligned}
$$

Calculate the case where $C = 0$,

$$
\begin{aligned}
P(R_1 = 1 \mid C = 0) &= \frac{0.01 + 0.05 + 0.09}{(0.01 + 0.05 + 0.09) + (0.99 + 0.95 + 0.91)} \\
&= 0.05 \\
P(R_1 = 0 \mid C = 0) &= 1 - 0.05 \\
&= 0.95
\end{aligned}
$$

Then using both cases,

$$
\begin{aligned}
P(R_1 = 1, R_2 = 0) &= \sum_{C} P(C = c)\tau_1(R_2 = 0 \mid C)\tau_2(R_1 = 1 \mid C) \\
&= (0.7)(0.95)(0.05) + (0.3)(0.8)(0.2) \\
&= 0.08125
\end{aligned}
$$

Hence,

$$
\begin{aligned}
P(C = 1 \mid R_1 = 1, R_2 = 0) &= \frac{P(R_1 = 1 \mid C = 1)P(R_2 = 0 \mid C = 1)P(C = 1)}{P(R_1 = 1, R_2 = 0)} \\
&= \frac{0.048}{0.08125} \\
&= \frac{192}{325} \\
&\approx 0.59
\end{aligned}
$$

3. (a) Implement the above Bayes net with the specified conditional probabilities into pgmpy. **[2.5 points]**

```
[30]
from pgmpy.models import BayesianNetwork
from pgmpy.factors.discrete import TabularCPD
import sys
```

```
[31]
# We first create a model which contains the edges of the graph
model = BayesianNetwork([('S', 'Y'),
                         ('S', 'C'),
                         ('A', 'C'),
                         ('A', 'I'),
                         ('I', 'C'),
                         ('R', 'C'),
                         ('F', 'R'),
                         ('P', 'R'),
                         ('R', 'B')])
```

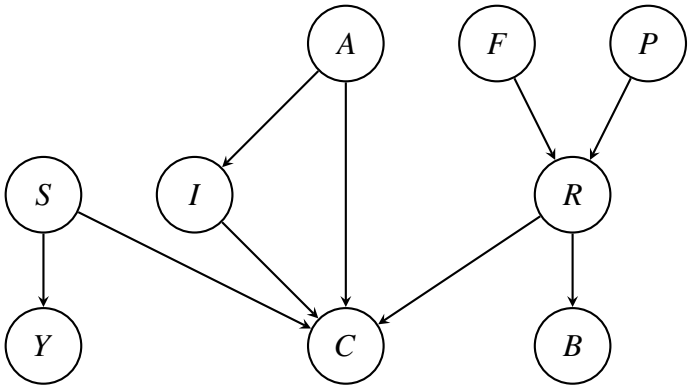
```
[32]
# Enter conditional probability distribution for each variable

# Prior probability for Smoking P(S)
cpd_S = TabularCPD(variable='S', variable_card=2, values=[[0.9], [0.1]])

# Prior probability for Alcohol Abuse P(A)
cpd_A = TabularCPD(variable='A', variable_card=2, values=[[0.9], [0.1]])

# Prior probability for Solar Flare P(F)
cpd_F = TabularCPD(variable='F', variable_card=2, values=[[0.999], [0.001]])

# Prior probability for Phone P(P)
cpd_P = TabularCPD(variable='P', variable_card=2, values=[[0.001], [0.999]])
```

```
[33]
# Conditional probability for Yellow fingers or P(Y | S)
cpd_Y = TabularCPD(variable='Y', variable_card=2,
                   evidence=['S'], evidence_card=[2],
                   values=[[0.96, 0.05], [0.04, 0.95]])

# Conditional probability for Weakened Immune System or P(I | A)
cpd_I = TabularCPD(variable='I', variable_card=2,
                   evidence=['A'], evidence_card=[2],
                   values=[[0.95, 0.7], [0.05, 0.3]])

# Conditional probability for Radiation or P(R | F, P)
cpd_R = TabularCPD(variable='R', variable_card=2,
                   evidence=['F', 'P'], evidence_card=[2, 2],
                   values=[[0.99, 0.95, 0.8, 0.7],
                           [0.01, 0.05, 0.2, 0.3]])

# Conditional probability for Burn Skin or P(B | R)
cpd_B = TabularCPD(variable='B', variable_card=2,
                   evidence=['R'], evidence_card=[2],
                   values=[[0.98, 0.8], [0.02, 0.2]])

# Conditional probability for Cancer or P(C | A, I, R, S)
cpd_C = TabularCPD(variable='C', variable_card=2,
                   evidence=['A', 'I', 'R', 'S'], evidence_card=[2, 2, 2, 2],
                   values=[[0.9, 0.5, 0.8, 0.4, 0.7, 0.4, 0.6, 0.2, 0.8, 0.4, 0.7, 0.3, 0.6, 0.3, 0.5, 0.1],
                           [0.1, 0.5, 0.2, 0.6, 0.3, 0.6, 0.4, 0.8, 0.2, 0.6, 0.3, 0.7, 0.4, 0.7, 0.5, 0.9]])
```

```
[34]
model.add_cpds(cpd_S, cpd_F, cpd_P, cpd_A, cpd_B, cpd_I, cpd_R, cpd_C, cpd_Y)
print(model.check_model())
```

```
...  True
```

(b) Draw the Bayesian network clearly showing the nodes and arrows showing relationship among all the variables. **[1.5 points]**



(c) What is the probability of radiation given cancer? Show values for $R \in \{0, 1\}$. **[1.5 points]**

$$P(R = 1 \mid C = 1)$$

```
from pgmpy.inference import VariableElimination
infer = VariableElimination(model)

# Get probability of Radiation given Cancer P(R=1 | C=1)
phi_query = infer.query(['R'], evidence={'C':1}, joint = False)
factor = phi_query['R']
print('Probability of Radiation given Cancer')
print(factor)

# Output
'''
Probability of Radiation given Cancer
+------+----------+
| R    |   phi(R) |
+======+==========+
| R(0) |  0.9214 |
+------+----------+
| R(1) |  0.0786 |
+------+----------+
'''
```

$$P(R = 0 \mid C = 1) = 0.9214$$
$$P(R = 1 \mid C = 1) = 0.0786$$

(d) What is the probability of cancer given the patient has skin burn, yellow fingers and abuses alcohol? Show values for $C \in \{0, 1\}$. **[1.5 points]**

$$P(C = 1 \mid B = 1, Y = 1, A = 1)$$

```
# Get probability of Cancer given skin burn, yellow fingers and alcohol abuse. P(C=1 | B=1, Y=1, A=1)
phi_query = infer.query(['C'], evidence={'B':1, 'Y':1, 'A':1}, joint = False)
factor = phi_query['C']
print('Probability of Cancer given skin burn, yellow fingers and alcohol abuse')
print(factor)

# Output
'''
Probability of Cancer given skin burn, yellow fingers and alcohol abuse
+------+----------+
| C    |   phi(C) |
+======+==========+
| C(0) |  0.4296 |
+------+----------+
| C(1) |  0.5704 |
+------+----------+
'''
```

$$P(C = 0 \mid B = 1, Y = 1, A = 1) = 0.4296$$
$$P(C = 1 \mid B = 1, Y = 1, A = 1) = 0.5704$$

(e) Are Smoking and skin burn independent given that cancer is present? Justify your answer. **[1.5 points]**
No. They are dependent.There is a V-structure between S, C and R.

$$S \rightarrow C \leftarrow R$$

Given C, it couples S and R. This makes S and R dependent. Since R and B are dependent due to cascade structure

$$R \rightarrow B$$

then S and B are dependent.

(f) What is the probability of cancer if you never abused alcohol or used a cellphone? **[1.5 points]**

$$P(C = 1 \mid A = 0, P = 0)$$

```
# Get probability of Cancer given no alcohol and no cellphone. P(C=1 | A=0, P=0)
phi_query = infer.query(['C'], evidence={'A':0, 'P':0}, joint = False)
factor = phi_query['C']
print('Probability of Cancer given no alcohol and no cellphone')
print(factor)

# Output
'''
```

```
        Probability of Cancer given no alcohol and no cellphone
        +------+----------+
        | C    |   phi(C) |
        +======+==========+
        | C(0) |  0.8495  |
        +------+----------+
        | C(1) |  0.1505  |
        +------+----------+
        '''
```

$$P(C = 0 \mid A = 0, P = 0) = 0.8495$$
$$P(C = 1 \mid A = 0, P = 0) = 0.1505$$

4. You can build your solution on top of the python notebook covered in class to classify the standard MNIST dataset.

   Full solution found in `Q4_template_MNISTCorrupted.ipynb`.

```
from matplotlib import pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers
import tensorflow_datasets as tfds

## write your code here
dataset_name = "mnist_corrupted/zigzag"
train_ds = tfds.load(dataset_name, split='train', batch_size=-1,
    as_supervised=True)
test_ds = tfds.load(dataset_name, split='test', batch_size=-1, as_supervised=True)

train_images, train_labels = tfds.as_numpy(train_ds)
test_images, test_labels = tfds.as_numpy(test_ds)

# Test size of different loaded numpy arrays
print('Image size:', train_images[0].shape)
print('Training data size:',train_images.shape)
print('Training labels size:', train_labels.shape)
print('Testing data size:', test_images.shape)

model = tf.keras.Sequential()
outputs = 10 #because there are 10 digits in mnist
## write your code here to build your dense ANN. Input layer is created below
model.add(layers.Flatten(input_shape=(train_images[0].shape)))
model.add(layers.Dense(10, activation=tf.nn.relu))
model.add(layers.Dense(20, activation=tf.nn.relu))
model.add(layers.Dense(20, activation=tf.nn.relu))
model.add(layers.Dense(60, activation=tf.nn.relu))
model.add(layers.Dense(60, activation=tf.nn.relu))
model.add(layers.Dense(80, activation=tf.nn.relu))
model.add(layers.Dense(80, activation=tf.nn.relu))
model.add(layers.Dense(100, activation=tf.nn.relu))
model.add(layers.Dense(100, activation=tf.nn.relu))
model.add(layers.Dense(80, activation=tf.nn.relu))
model.add(layers.Dense(80, activation=tf.nn.relu))
model.add(layers.Dense(60, activation=tf.nn.relu))
model.add(layers.Dense(60, activation=tf.nn.relu))
model.add(layers.Dense(20, activation=tf.nn.relu))
model.add(layers.Dense(20, activation=tf.nn.relu))
model.add(layers.Dense(10, activation=tf.nn.softmax))
model.summary()

### write your code here to compile model
model.compile(optimizer="Adam", loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
```

```
45
46  ### write your code here to train your model
47  epochs = 10
48  history = model.fit(train_images, train_labels, epochs=epochs)
49
50  plt.plot(history.history["loss"])
51
52  #### write your code to report overall accuracy on test set
53  test_results = model.evaluate(test_images, test_labels, return_dict=True)
54  # print(test_results)
55  print('Test accuracy:', test_results['accuracy'])
56
57  ### write your code to report per-class accuracy
58  ### Use confusion matrix from sklearn.
59  from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
60  image_pred = model.predict(test_images)
61  image_pred_classes = image_pred.argmax(axis=-1)
62  labels = [0,1,2,3,4,5,6,7,8,9]
63  cm = confusion_matrix(test_labels, image_pred_classes, labels=labels)
64  disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
65  disp.plot()
66
67  import matplotlib.pyplot as plt
68  %matplotlib inline
69  class_names = ['0','1','2','3','4','5','6','7','8','9']
70  plt.figure(figsize=(10,10))
71  for i in range(25):
72      plt.subplot(5,5,i+1)
73      plt.xticks([])
74      plt.yticks([])
75      plt.grid(False)
76      plt.imshow(train_images[i].reshape(28, 28), cmap=plt.cm.binary)
77      #print(train_labels[i][0])
78      plt.xlabel(class_names[train_labels[i]])
```

5. In this question, we will learn how to use transfer learning in the context of CNNs.

   (a) Create the LeNet-5 CNN architecture using Keras API (see code skeleton for the number and types of layers to create). Train the model on the MNIST dataset. **[3 points]**

```
input_shape = train_images[0].shape
model = tf.keras.Sequential()
model.add(layers.Conv2D(6, kernel_size=(5, 5), strides=(1, 1),
    activation='tanh', padding='same', input_shape=input_shape))
model.add(tf.keras.layers.AveragePooling2D(pool_size=(2, 2),
    strides=None, padding='valid', data_format=None))
model.add(layers.Conv2D(16, kernel_size=(5, 5), strides=(1, 1),
    activation='tanh', padding='valid'))
model.add(tf.keras.layers.AveragePooling2D(pool_size=(2, 2),
    strides=None, padding='valid', data_format=None))
model.add(layers.Conv2D(120, kernel_size=(5, 5), strides=(1, 1),
    activation='tanh', padding='valid'))
model.add(layers.Flatten())
model.add(layers.Dense(84, activation='tanh'))
model.add(layers.Dense(10, activation='softmax'))
model.summary()

# Compile the model with appropriate Loss function
model.compile(optimizer=tf.optimizers.Adam(learning_rate=0.001),
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])

# Train the model on MNIST dataset
epochs = 5
batch_size = 512
model.fit(train_images, train_labels, batch_size=batch_size,
    epochs=epochs)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc) #0.97
```

   (b) What is the accuracy of your trained LeNet-5 model on the MNIST training dataset? Try to get an accuracy above 90%. **[0.5 points]**

Test Accuracy = 0.9735

   (c) Download the `binary_alpha_digits` dataset using tfds, and split the dataset into 20% testing data and 80% training data.**[1 point]**

```
## write your code here
dataset_name = "binary_alpha_digits"
train_ds, test_ds = tfds.load(dataset_name,
    split=['train[0\%:80\%]','train[80\%:100\%]'], batch_size=-1,
    as_supervised=True)

train_images, train_labels = tfds.as_numpy(train_ds)
```

```
        test_images, test_labels = tfds.as_numpy(test_ds)

        print('Image size:', train_images[0].shape)
        print('Training data size:',train_images.shape)
        print('Training labels size:', train_labels.shape)
        print('Testing data size:', test_images.shape)
```

(d) As the dimension of images in the `binary_alpha_digits` are different from the image size in MNIST dataset, upscale images in `binary_alpha_digits` to match the image size in MNIST dataset using OpenCV. This is required as we would like to use the LeNet trained using the MNIST dataset for `binary_alpha_digits`. **[2 points]**

```
        newSize = 28
        # create a numpy array for storing upscaled training images
        train_upscale = np.zeros((train_images.shape[0], newSize, newSize, 1))
        for i, t in enumerate(train_images):
          resized = cv2.resize(t, (newSize, newSize))
          train_upscale[i] = resized.reshape(newSize, newSize, 1)
        print("Train upscale shape: ",train_upscale.shape)
        print("Train images shape: ", train_images.shape)

        # create a numpy array for storing upscaled testing images
        test_upscale = np.zeros((test_images.shape[0], newSize, newSize, 1))
        for i, t in enumerate(test_images):
          resized = cv2.resize(t, (newSize, newSize))
          test_upscale[i] = resized.reshape(newSize, newSize, 1)

        print("Test upscale shape: ", test_upscale.shape)
        print("Test images shape: ", test_images.shape)
```

(e) Remove the final output layer of LeNet you have trained on MNIST (to do this, please check the flag "include top" in Keras and the tensorflow link for transfer learning noted earlier) **[0.5 points]**

```
        transfer_model = tf.keras.Sequential()
        for layer in model.layers[:-3]: #remove last three layers
          transfer_model.add(layer)

        #Freeze layers
        transfer_model.get_layer('conv2d').trainable = True

        print(f"All layers: {list(layer.name for layer in
            transfer_model.layers)}.")
        print(f"Frozen layers: {list(layer.name for layer in
            transfer_model.layers if (not
            transfer_model.get_layer(layer.name).trainable))}.")
```

(f) After removing the final output layer, extend your trained LeNet model by adding at least one hidden layer (dense, convolution, max pooling or any other type of layer). Also attach one final output layer. In this part, you are free to explore and decide how many hidden layers to add, their type, the number of nodes in each layer and the activation function yourself. Keep in mind, the

output layer must have the appropriate number of nodes and activation function that matches the given task. **[1.5 points]**

```
transfer_model.add(layers.Flatten())
transfer_model.add(layers.Dense(84, activation='tanh'))
transfer_model.add(layers.Dense(64, activation='tanh'))
transfer_model.add(layers.Dense(36, activation='softmax')) #36 classes
transfer_model.summary()
```

(g) Train the model and show accuracy on the testing dataset (of `binary_alpha_digits`). You can either fix all the weights of your MNIST-trained LeNet model and train only the layers you have added, or train the whole network again. Choose the setting that gives you higher accuracy given the computational resources. Check link https://keras.io/getting-started/faq/#how-can-i-freeze-keras-layers. Try to achieve a testing data accuracy of 50% or more (you can report the best over multiple runs). Please make sure that in your submitted jupypter notebook, logs show your best run. Note: some variation between runs is expected, with the true accuracy being somewhere in between. You are not required to reliably get 50 percent accuracy over all runs, but try to demonstrate from the log files that one run achieved 50 percent. **[1.5 points]**

```
transfer_model.compile(optimizer=tf.optimizers.Adam(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])

epochs = 40
batch_size = 512
transfer_model.fit(train_upscale, train_labels, batch_size=batch_size,
    epochs=epochs)

test_loss, test_acc = transfer_model.evaluate(test_upscale, test_labels)
print('Test accuracy:', test_acc)
```

Test Accuracy = 0.7224