CAP6619 Assignment #2 Question 7

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.utils import shuffle
import tensorflow as tf
from tensorflow import keras
print(tf.__version__)

from os import listdir
from numpy import asarray, save
from tensorflow.keras.utils import load_img, img_to_array
from tensorflow.keras.layers import BatchNormalization
from keras.layers import Activation, Flatten, Dense, Dropout
from tensorflow.keras import layers
from matplotlib.image import imread
```

```
2.10.0
```

In [ ]:
```
# get dataset
folder = 'dogvscat1000/'
photos, labels = list(), list()
for file in listdir(folder):
    output = 0.0
    if file.startswith('cat'):
        output = 1.0
    photo = load_img(folder + file, target_size=(32,32), color_mode='rgb')
    photo = img_to_array(photo)
    photos.append(photo)
    labels.append(output)
photos = asarray(photos)
labels = asarray(labels)
print(photos.shape, labels.shape)
save('dogs_vs_cats_photos.npy', photos)
save('dogs_vs_cats_labels.npy', labels)
```

```
(1000, 32, 32, 3) (1000,)
```

In [ ]:
```
print(photos[1].shape)
```

```
(32, 32, 3)
```

In [ ]:
```
# Create training and test datasets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(photos, labels, test_size=0.4,
```

In [ ]:
```
# Use 10-fold cross validation to split the 1000 images into training vs test
from sklearn.model_selection import KFold
kf = KFold(n_splits=10, shuffle=True)
kf.get_n_splits(photos)
Acc, Acc2 = [], []
```

In [ ]:
```
from sklearn.metrics import accuracy_score
```

In [ ]:
```
# Create feedforward NN
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(32,32,3), name='Input'),
    keras.layers.Dense(50, activation='sigmoid', name='HiddenLayer1'),
    keras.layers.Dense(2, name='Output'),
])
```

In [ ]:
```
# Create feedforward NN with batchnorm and/or dropout
model2 = keras.Sequential([
    keras.layers.Flatten(input_shape=(32,32,3), name='Input'),
    keras.layers.Dense(100, activation='sigmoid', name='HiddenLayer1'),
    keras.layers.Dropout(rate=0.5),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(2, name='Output'),
])
```

In [ ]:
```
# function to conduct cross validation of a given model
for train_index, test_index in kf.split(photos):
    X_train, X_test = photos[train_index], photos[test_index]
    y_train, y_test = labels[train_index], labels[test_index]
    model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
    model2.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
    history = model.fit(X_train, y_train, epochs=100, verbose=1)
    history2 = model2.fit(X_train, y_train, epochs=100, verbose=1)
    y_pred=model.predict(X_test)
    y_pred2=model2.predict(X_test)
    y_pred=np.argmax(y_pred,axis=1)
    y_pred2=np.argmax(y_pred,axis=0)
    Acc.append(accuracy_score(y_test, y_pred))
    Acc2.append(accuracy_score(y_test, y_pred))
```

In [ ]:
```
# Show accuracy of the models
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print('\nTest accuracy:', test_acc)

test_loss, test_acc = model2.evaluate(X_test, y_test, verbose=2)
print('\nTest accuracy:', test_acc)
```

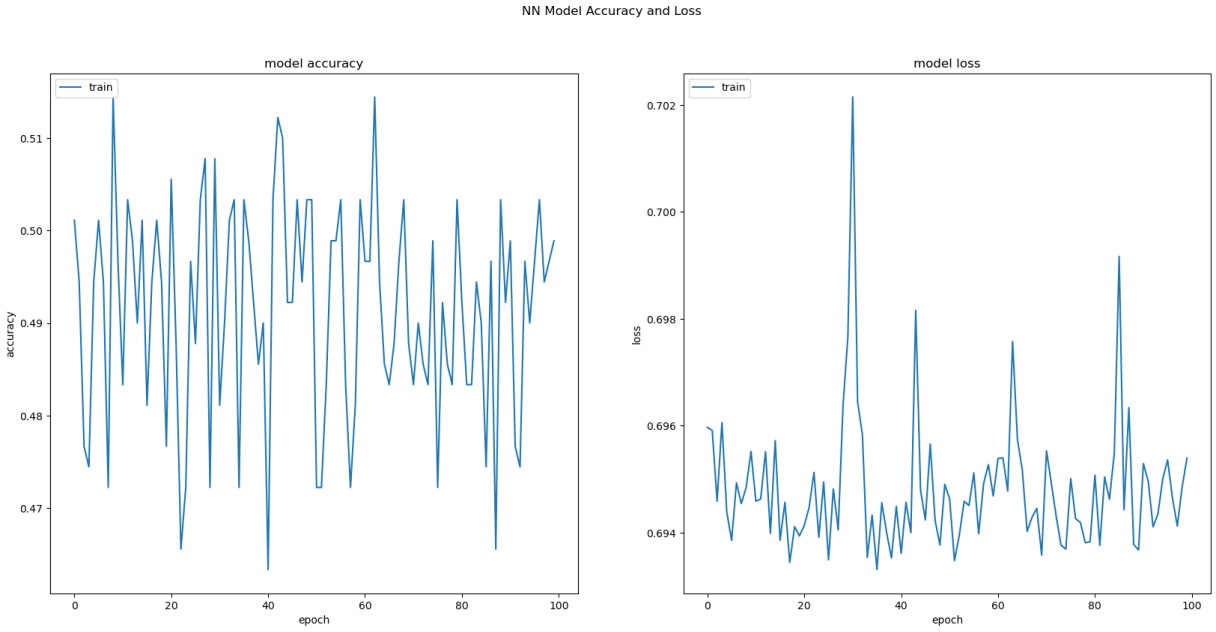
```
4/4 - 0s - loss: 0.7004 - accuracy: 0.4700 - 179ms/epoch - 45ms/step

Test accuracy: 0.4699999988079071
4/4 - 0s - loss: 0.6939 - accuracy: 0.4700 - 240ms/epoch - 60ms/step

Test accuracy: 0.4699999988079071
```

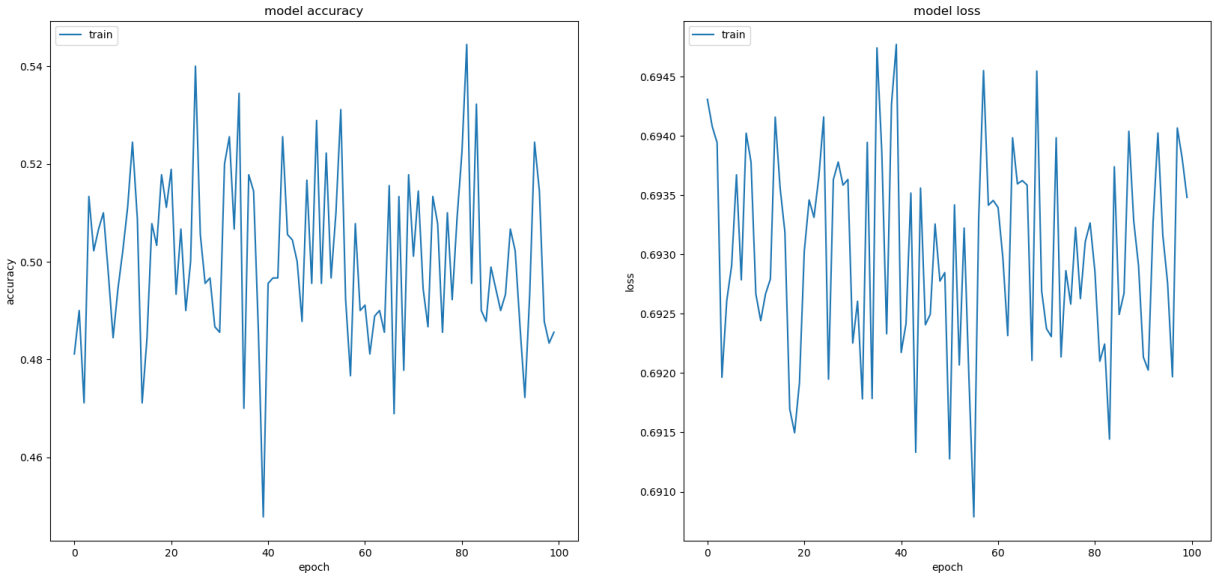
In [ ]:
```
# NN Model 1 Loss and Accuracy
fig, (ax1,ax2) = plt.subplots(nrows=1, ncols=2, figsize=(20,9))
ax1.plot(history.history['accuracy'])
ax1.set_title('model accuracy')
```

```
ax1.set_xlabel('epoch')
ax1.set_ylabel('accuracy')
ax1.legend(['train', 'validation'], loc='upper left')
ax2.plot(history.history['loss'])
ax2.set_title('model loss')
ax2.set_xlabel('epoch')
ax2.set_ylabel('loss')
ax2.legend(['train', 'validation'], loc='upper left')
fig.suptitle('NN Model Accuracy and Loss')
plt.show()
```



In [ ]:
```
# NN Model 2 (with BatchNorm and Dropout) Loss and Accuracy
fig, (ax1,ax2) = plt.subplots(nrows=1, ncols=2, figsize=(20,9))
ax1.plot(history2.history['accuracy'])
ax1.set_title('model accuracy')
ax1.set_xlabel('epoch')
ax1.set_ylabel('accuracy')
ax1.legend(['train', 'validation'], loc='upper left')
ax2.plot(history2.history['loss'])
ax2.set_title('model loss')
ax2.set_xlabel('epoch')
ax2.set_ylabel('loss')
ax2.legend(['train', 'validation'], loc='upper left')
fig.suptitle('NN Model (with BatchNorm and Dropout) Accuracy and Loss')
plt.show()
```

NN Model (with BatchNorm and Dropout) Accuracy and Loss



In [ ]: