Physics Dataset Practice Problems

Week of August 3, 2020

3 Colorectal Cancer Histology

3.1 Data Visualization

```
import tensorflow as tf
from PIL import Image
import matplotlib.pyplot as plt
import os
import numpy as np

data_directory = './histology/'
classes = os.listdir(data_directory)
fig, axs = plt.subplots(1,8)
for i in range(8):
    dir = os.path.join(data_directory, classes[i])
    file = os.listdir(dir)[0]
    image = Image.open(os.path.join(dir, file))
    #print(np.amax(image)) #get max for normalization
    axs[i].imshow(image)
    axs[i].set_title(classes[i])
    axs[i].axis('off')
plt.show()
```

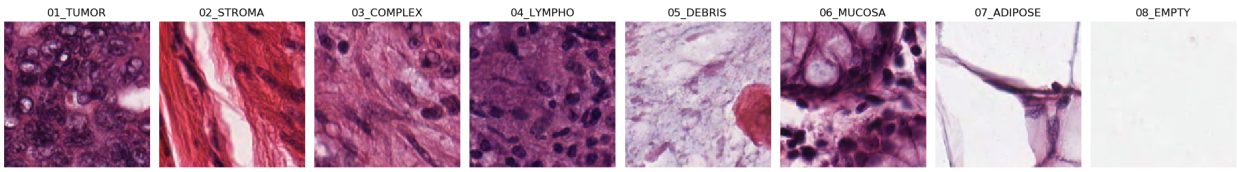

Figure 1: Some examples from the histology dataset.

3.2 Data Preprocessing

```
#Use data generators to preprocess and augment image data
datagenerator = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255,
                                    rotation_range=40,
                                    width_shift_range=0.2,
                                    height_shift_range=0.2,
                                    shear_range=0.2,
                                    horizontal_flip=True,
                                    zoom_range=0.2,
                                    fill_mode='nearest',
                                    validation_split=0.2)

# Load images in batches of 20 using flow_from_directory method
train_generator = datagenerator.flow_from_directory(data_directory,
                                                    target_size=(75,75),
```

```
                                                    batch_size=64,
                                                    class_mode='categorical',
                                                    subset='training')

validation_generator = datagenerator.flow_from_directory(data_directory,
                                                         target_size=(75,75),
                                                         batch_size=64,
                                                         class_mode='categorical',
                                                         subset='validation')
```

3.3 Initialize a Model

```
#Initialize a model with tf.keras
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), input_shape=(75,75,3), activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(8, activation='softmax')
])
```

3.4 Train the Model

```
model.compile(optimizer=tf.keras.optimizers.SGD(lr=0.001, momentum=0.9),
        metrics=['accuracy'],
        loss='categorical_crossentropy')

history = model.fit(train_generator, validation_data=validation_generator, epochs=50)
```

```
63/63 [==============================] - 43s 681ms/step - loss: 1.8143 - accuracy: 0.2355 - val_loss: 1.6169 - val_accuracy: 0.4950
Epoch 2/50
63/63 [==============================] - 41s 655ms/step - loss: 1.3837 - accuracy: 0.3887 - val_loss: 1.3111 - val_accuracy: 0.5250
Epoch 3/50
63/63 [==============================] - 41s 648ms/step - loss: 1.1597 - accuracy: 0.4660 - val_loss: 1.2475 - val_accuracy: 0.4210
Epoch 4/50
63/63 [==============================] - 41s 651ms/step - loss: 1.0924 - accuracy: 0.5013 - val_loss: 1.1357 - val_accuracy: 0.5000
Epoch 5/50
28/63 [============>.................] - ETA: 18s - loss: 1.0147 - accuracy: 0.5513
```

Figure 2: I trained my model for 50 epochs with a learning rate of 0.001.

```
#plot the model performance during training
#print(history.history.keys()) #i forgot what the keys are called
loss = history.history['loss']
val_loss = history.history['val_loss']
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

fig, axs = plt.subplots(1,2)
axs[0].plot(loss, label='Training')
axs[0].plot(val_loss, label='Validation')
axs[0].set_ylabel('CCE Loss')
axs[0].set_xlabel('Epoch')
```

```
axs[0].legend()
axs[1].plot(acc, label='Training')
axs[1].plot(val_acc, label='Validation')
axs[1].set_ylabel('Classification Accuracy')
axs[1].set_xlabel('Epoch')
axs[1].legend()
fig.suptitle('Model Performance During Training')
plt.show()
```

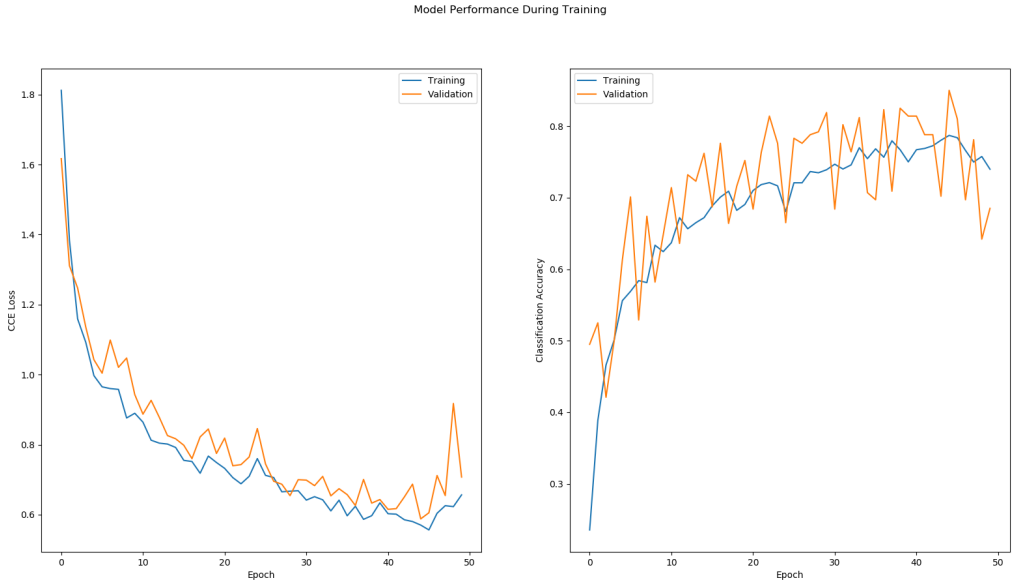

Figure 3: The loss decreases and accuracy increases for each dataset during training.

```
(images, targets) = next(validation_generator)
images = images[0:8]; targets = targets[0:8]
preds = model.predict(images)
preds = np.argmax(preds, axis=1)
targets = np.argmax(targets, axis=1)
print(preds.shape)

fig, axs = plt.subplots(1,8)
for i in range(8):
    dir = os.path.join(data_directory, classes[i])
    file = os.listdir(dir)[0]
    image = Image.open(os.path.join(dir, file))
    #print(np.amax(image)) #get max for normalization
    axs[i].imshow(image)
    axs[i].set_title('True: ' + str(classes[targets[i]]))
    axs[i].set_xlabel('Pred: ' + str(classes[preds[i]]))
    axs[i].set_xticks([])
    axs[i].set_yticks([])
plt.show()
```

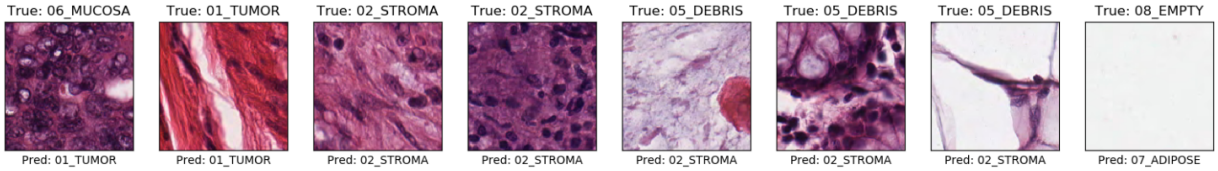

Figure 4: Despite a few errors, the model performs pretty well.