"DeepLearning.AI TensorFlow Developer" Specialization

Andrii X

1 Introduction to TensorFlow for AI, ML, and DL

1.1 Week 1

- **Simple example aka "Hello, World!":**
  Define NN (1 layer with 1 neuron):

```
# Build a simple Sequential model
model = tf.keras.Sequential(
[keras.layers.Dense(units=1, input_shape=[1])]
)
```

  Compile the model:

```
model.compile(optimizer='sgd', loss='mean_squared_error')
```

  Provide the data:

```
# Declare model inputs and outputs for training
xs = np.array([-1.0,  0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
```

  Train the NN:

```
# Train the model
model.fit(xs, ys, epochs=500)
```

  Use trained NN for new data:

```
# Make a prediction
print(model.predict([10.0]))
```

1.2 Week 2

- **A Computer Vision Example: Fashion MNIST dataset**
  The Fashion MNIST dataset is a collection of grayscale 28x28 pixel clothing images.
  1) Load the Fashion MNIST dataset:

```
fmnist = tf.keras.datasets.fashion_mnist
```

  2) Load the training and test split of the Fashion MNIST dataset:

```
(training_images, training_labels),
(test_images, test_labels) = fmnist.load_data()
```

  3) Normalize the pixel values of the train and test images:

```
training_images  = training_images / 255.0
test_images = test_images / 255.0
```

  4) Build the classification model:

```
model = tf.keras.models.Sequential([
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(128, activation=tf.nn.relu),
tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

  Sequential - defines a sequence of layers in the neural network. **Flatten** - converts a 28x28 matrix into a 1-D array. **Dense** - adds a layer of neurons. Activation function **relu** passes values greater than 0 to the next layer. **Softmax** takes a list of values and scales these so the sum of all elements will be equal to 1. When applied to model outputs, you can think of the scaled values as the probability for that class.
  5) Compile and train the model:

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

model.fit(training_images, training_labels, epochs=5)
```