**Deep Learning with TensorFlow**

**Section 1: Getting Started**

**1.1 The Course Overview**

This video gives an overview of the entire course.

**1.2 Installing TensorFlow**

TensorFlow is a new machine library that is probably not installed on our operating system by default. This video will guide you through installing TensorFlow locally or remotely.

* + Note TensorFlow as a new machine learning library with documentation
  + Install TensorFlow locally via pip
  + Discuss how SageMathCloud supports TensorFlow

**1.3 Simple Computations**

Before we can use TensorFlow for deep learning, we need to understand how TensorFlow handles basic objects and operations. This video will walk you through a few computations.

* + Understand TensorFlow objects as multidimensional “tensors”
  + Describe TensorFlow computation as specifying a graph
  + Execute all or part of a graph using a “session”

**1.4 Logistic Regression Model Building**

Learning any library from documentation can be challenging, so we're going to build a practical machine learning classifier with TensorFlow. We'll start with a simple logistic regression classifier and build up from there.

* + We will classify images of characters by font
  + Logistic regression is a simple way to “score” possible fonts
  + We will quickly code the entire classifier in TensorFlow

**1.5 Logistic Regression Training**

Though we have a classifier, we need to compute weights so that our model is accurate. For this, we can use TensorFlow to specify and optimize a loss function. TensorFlow will then use this to find good weights.

* + Build a TensorFlow “categorical cross-entropy” function to optimize weights
  + Train the model with gradient descent over many epochs
  + Evaluate model accuracy and weights

**Section 2: Deep Neural Networks**

**2.1 Basic Neural Nets**

Using single pixels as features limits us to model essentially linear phenomena. To model non-linear things such as font styles involving several pixels, we will use neural networks to transform our inputs into non-linear combinations for use in a logistic regression classifier.

* + Explain simple non-linear transformations of values with activation functions
  + Implement a single neuron in TensorFlow
  + Describe the structural nature of a neural network

**2.2 Single Hidden Layer Model**

Now that we understand neural networks, we can see how TensorFlow makes them easy to implement and train.

* + Implement the neural net in TensorFlow
  + Briefly describe how backpropagation trains the weights
  + Use TensorFlow to train the neural net

**2.3 Single Hidden Layer Explained**

Now that we've trained a neural network, we should inspect it closely to understand the accuracy and weights.

* + Let's examine the test accuracy over many epochs of training
  + Create a confusion matrix to understand model errors
  + Inspect and visualize the learned weights

**2.4 Multiple Hidden Layer Model**

A single hidden layer is good, but you may find the number of neurons growing prohibitive in order to model very complex features. To combine features more easily, we expand the network in depth rather than width. It's true deep learning with multiple hidden layers.

* + Instead of adding more neurons, let's add a new layer.
  + Deciding the number of neurons and layers is a hard experiment!
  + Train the deep neural network. This may take some time.

**2.5 Multiple Hidden Layer Results**

With our deep neural net trained, we should take time to check its accuracy and understand the features it's extracting.

* + Verify model accuracy on training and testing data
  + Visualize and understand the weights of the input layer
  + Visualize and understand the weights of the output layer

**Section 3: Convolutional Neural Networks**

**3.1 Convolutional Layer Motivation**

Particularly in images, the features that we want to find can occur anywhere among the pixels. Convolutional neural nets allow us to train one set of weights to search small windows of an image for a feature.

* + Understand convolution as sliding windows with one set of weights
  + Sets of weights allow pulling of many features from one window
  + Handle multiple pixel channels (like RGB colors)

**3.2 Convolutional Layer Application**

Understanding the theory of convolutional layers is useless without learning the tools to actually use them. This video walks through a simple example with TensorFlow.

* + Describe an example setup and unusual shapes of variables
  + Implement weights and call TensorFlow convolution code
  + Evaluate sample node and observe the effect of convolution

**3.3 Pooling Layer Motivation**

Convolutions can find a feature anywhere in an image, but with all the overlap, we need to make sure we don't find the same feature in the same place multiple times. A pooling layer reduces the size of our input, taking only relevant information.

* + Max pooling is a sliding window without overlap
  + Poolings usually don't consider partial windows
  + Combine convolutional and pooling layers for a maximum effect

**3.4 Pooling Layer Application**

Having learned how Max Pooling works in theory, it's time to put it into practice by adding it to our simple example in TensorFlow.

* + Tf.maxpool is the core code call
  + For a final pooling layer, we should flatten the output.
  + Confirm that 2x2 max pooling has cut the shape into half

**3.5 Deep CNN**

We've learned about convolutional layers and used them in an example, but now let's use them for real by adding a convolutional layer to the font classification model.

* + Setup window size and weights with correct shape
  + Implement convolution and pooling layers. Flatten the parameters.
  + Evaluate the model accuracy

**3.6 Deeper CNN**

Convolutional layers often work well when chained together. Let's add another to our font classification model.

* + Set up a new convolutional layer
  + Carefully transition from convolutional to dense layer
  + Describe dropout training to avoid overfitting

**3.7 Wrapping Up Deep CNN**

After our deep model has trained for a model, it's time to see how well it performs.

* + Note that there are no dropouts during testing
  + Observe the model accuracy
  + Understand convolutional weights as small image features

**Section 4: Recurrent Neural Networks**

**4.1 Introducing Recurrent Neural Networks**

Some problems have time-based inputs. Features from the recent past might matter to the current prediction. To address these, researchers have developed recurrent neural networks. TensorFlow natively supports these.

* + Explain the background of recurrent neural networks
  + Describe the example problem of predicting the season from the weather
  + Implement a season predictor in TensorFlow with a recurrent neural network

**4.2 skflow**

TensorFlow models can be cumbersome to specify, yet follow a common pattern. skflow provides a simple interface for typical models.

* + Introduce skflow as sklearn for TensorFlow
  + Build a simple font classification model with skflow
  + Build a custom font classification model with skflow

**4.3 RNNs in skflow**

RNNs can be hard to specify, but skflow will let us quickly build a model.

* + Reshape data for skflow RNN
  + One-line model specification
  + Train and evaluate

**Section 5: Wrapping Up**

**5.1 Research Evaluation**

After learning so many methods and building all these models, it's helpful to look back and see how far we've come.

* + Revisit font image data
  + Review simple and densely connected models
  + Review convolutional models

**5.2 The Future of TensorFlow**

TensorFlow is changing very quickly and is being adopted by more researchers and professionals. But at its core are the contributions submitted by new users.

* + Note the fast pace of development in TensorFlow
  + Describe the generality of TensorFlow
  + Encourage viewers to contribute to and improve TensorFlow