FUNDAMENTALS OF DEEP LEARNING FOR MULTIPLE DATA TYPES

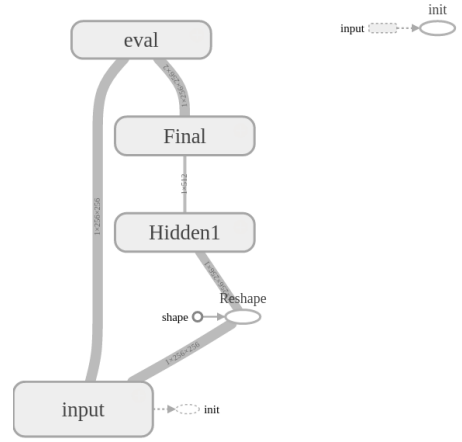
By:
Mohamed Aziz Tousli
Saifeddine Barkia

TASK 1 - SEGMENTATION

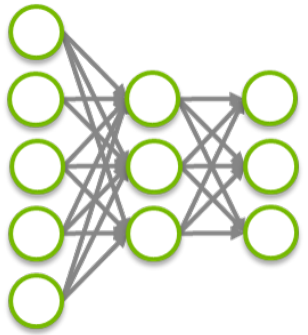
Input:
An image of size (x,y)

Output:
A tensor of size (x,y,n), n is the number of classes to be segmented

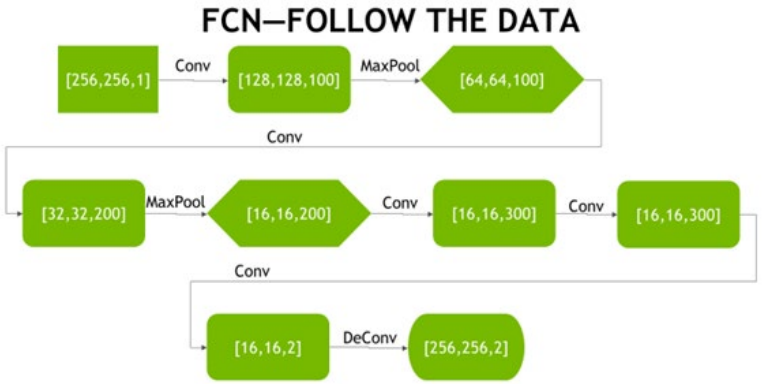
- Image segmentation = Semantic segmentation: Place each pixel into a specific class
  → Classification problem where you'll classify on a pixel basis rather than an entire image
- TensorFlow: The computations are expressed as data **flow** graphs which operate on **tensors**
- TFRecords: A special file format provided by TF, allowing you to use built-in TF functions
- TensorFlow basics:
  1. **Data flow graph**: Prepare the computations that will be performed
     → Variable = Weight = A parameter to be "learned"
  2. **Session**: Computations will be performed
     → Specify input data and training parameters to your previously-constructed graph
- Sample workflow:
  1. Prepare input data in TFRecords format
  2. Build the computation graph
  3. Train the model: inject input data into the graph in a TensorFlow **Session** and loop over your input data
  4. Evaluate the model: run **inference** (using the same graph from training) on previously unseen data and evaluate the accuracy of your model based on a suitable metric.
- PS: !Instruction #Bash instruction in Jupyter Notebook
- TensorBoard: Feature to visualize the representation of the computation graph



- One hidden layer: Create, train and evaluate a fully-connected NN with one hidden layer
  - sparse_softmax_cross_entropy_with_logits() #Combines softmax with cross entropy to calculate the loss
  - tf.name_scope() #Name a particular scope of the program → Code organization
  - tf.Variable() #Indicate a variable that will be trained (tensor of weights)



- Convolutional Neural Network:
  - Utility: Capture small regions of interest and large receptive fields
  - Pooling layers: Down-sample the data while attempting to retain most of the information
  - Deconvolution = Upscore: Bring a smaller image data set back up to its original size for final pixel classification
  - Fully convolutional network FCN: The result of modifying a CNN to adapt it to segmentation
  - Imbalance: When one class is much more probable than the other class



- CNN with Dice Metric for Accuracy:
  - Dice metric = Sorensen-Dice coefficient: $\frac{2A_{nl}}{A_n+A_l}$

A_n the area of the contour predicted by the network
A_l the area of the contour from the expertly-segmented label
A_{nl} the area of the contour that is predicted correctly by the network, i.e, the intersection of An and Al

- Parameter search:
  - learning_rate: the initial learning rate
  - decay_rate: the rate that the initial learning rate decays., e.g., 1.0 is no decay, 0.5 means cut the decay rate in half each step, etc.
  - decay_steps: the number of steps to execute before changing the learning rate
- Further enhancements:
  - Run Training Longer -- We ran very short training runs but in reality we'd run many more epochs.
  - More Training Data -- We only had 236 images in our training set. We could gather more data and we could also augment the data we have. TensorFlow has built-in functions to flip/rotate/transpose images automatically.
  - Larger networks -- We could try using AlexNet or other large CNN and convert them to FCN.

TASK 2 - WORD GENERATION

Number of lines = Number of words in the sentence
Number of columns = Number of words in the dictionary
Number of matrices = Number of sentences (1 matrix = 1 sentence)

- Data Preparation:
  - One hot encoding:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \Leftrightarrow$$

	This			
Is				
		An		
				Example
			EOS	

- The Word Generation Workflow:
  - Recurrent Neural Network (RNN): Structured to "remember" the words that led to their prediction.
- Improving performance:
  - Dropout: "Forget" some parameters when training → Reduce overfitting
- Train an RNN with MSCOCO Captions:
  - MSCOCO = Microsoft Common Objects in Context
  - Tokenize: Convert each word into a number (in descending order based on popularity)
- Image captioning: Describing an image
- LSTM: Long Short Term Memory