**Note: Much of the summary below was referenced from:**

@book{atienza2018advanced,

title={Advanced Deep Learning with Keras},

author={Atienza, Rowel},

year={2018},

publisher={Packt Publishing Ltd}

}

**What is Keras?**

* **“Keras is a high-level library that sits on top of other deep learning models.”**
* **Keras is supported on CPU, GPU, and TPU.**

TensorFlow, CNTK, MXNet, Theano

CPU

GPU

TPU

Keras

**“Backends”**

**Processor**

**Types**

**Supported**

**Side Note: What are TPUs?**

* **TPU = Tensor Processing Unit**
* **These are AI accelerator application-specific integrated circuits (ASIC). Initially they were created by Google. They were designed to train neural networks with TensorFlow.**
* **You can get clusters of TPUs currently from one cloud provider, Google Cloud, to train models (“Cloud TPUs”).**
* **There are also TPUs can be used to deploy models in edge computing situations (e.g. Intel Neural Compute Stick 2)**

**The Keras Sequential API**

* **The Keras Sequential Model API allows us to build and create models very quickly (if you understand what you are doing).**
* [**https://keras.io/getting-started/sequential-model-guide/**](https://keras.io/getting-started/sequential-model-guide/)
* **“Layers connect like Lego pieces”**
* **Model training (at minimum) only requires:**
  + **Data**
  + **A chosen number of training epochs**
  + **Evaluation metrics to monitor performance**
* **However, Keras can also be used for creating more advanced models with its “Model” and “Layer” classes.**
  + **This is achieved through the Functional API (**[**https://keras.io/getting-started/functional-api-guide/**](https://keras.io/getting-started/functional-api-guide/)**)**

**What are Tensors and Why Do They Flow?**

* **In deep learning, data is stored in tensors.**
* **What is a Tensor?** 
  + **Well that depends on the dimension of the tensor!**

|  |  |
| --- | --- |
| **Common Name** | **Type of Tensor** |
| Scalar | 0D tensor |
| Vector | 1D tensor |
| Matrix | 2D tensor |
| Multidimensional Tensor | 3D or greater tensor |

* **In other words, you often know what tensors are already, you’ve just called them something else.**

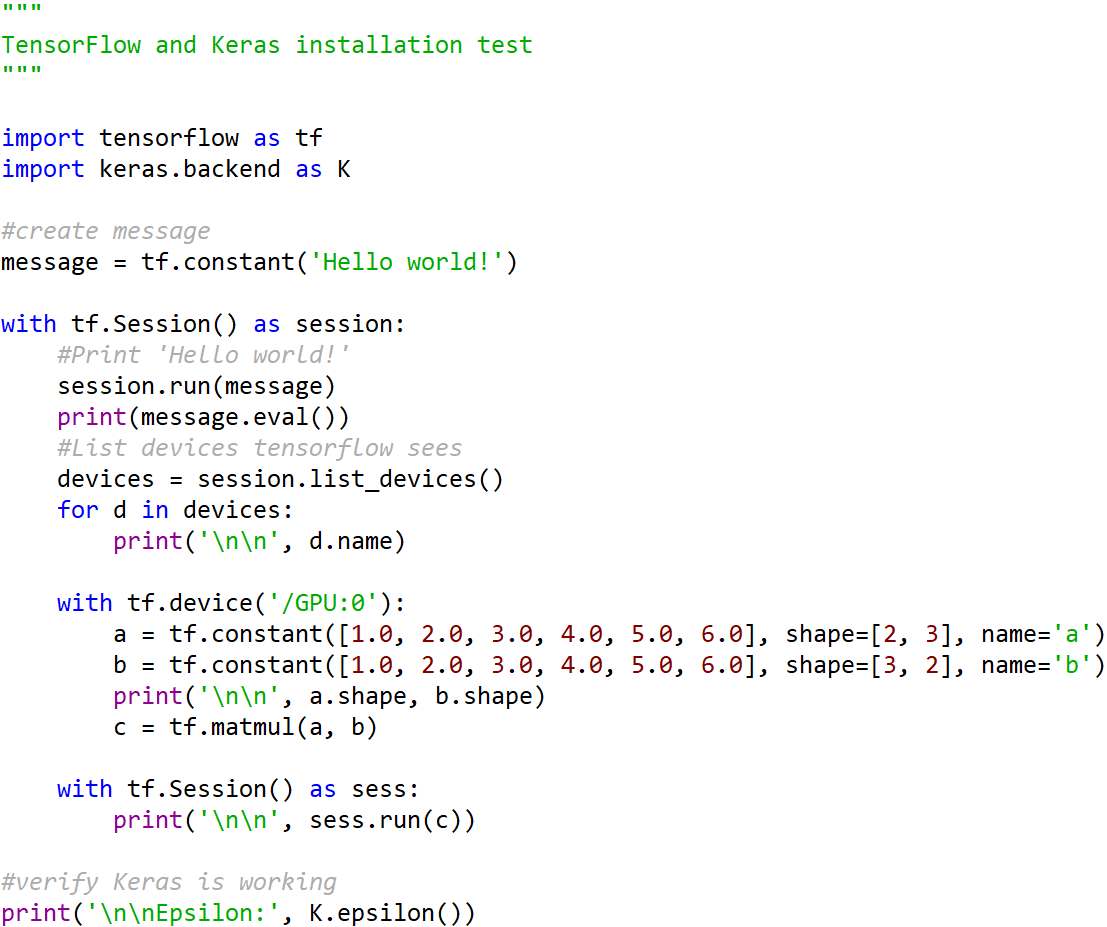
**Neural Network 3 Letter Acronyms: MLPs, CNNs, RNNs, GANs**

* **MLPs: Multilayer Perceptrons**
  + **Also called a “fully-connected network” or “feedforward neural networks”**
  + **Typically applied to logistic regression or linear regression problems**
  + **MLPs are not optimal for processing sequential and multi-dimensional data patterns**
* **RNNs: Recurrent Neural Networks**
  + **Good for sequential/time series data.**
  + **Find dependencies in historical data.**
* **CNNs: Convolutional Neural Networks**
  + **Great for processing multi-dimensional data (e.g. images and videos).**
  + **Extracts feature maps for classification, segmentation, and generation purposes.**
* **Often MLPs, RNNs, and CNNs are combined to create architectures for deep learning models.**
* **GANs: Generative Adversarial Networks**
  + **GANs can be used to generate new and meaningful outputs given arbitrary encodings**
  + **GANs learn how to model the input distribution by training two competing networks referred as the “generator” and the “discriminator”**
    - **Generator – discovers new fake data (could be images, audio, or video) to trick the discriminator.**
    - **Discriminator – trained to determine between real and fake signals in the data.**
    - **Once the discriminator can no longer see the difference between synthetically generated data and real data, it is discarded, and the generator is used to create realistic data.**

**Installing TensorFlow and Keras**

* **Tensorflow requires Python 3.4, 3.5, or 3.6 currently.**
  + [**https://www.tensorflow.org/install/pip**](https://www.tensorflow.org/install/pip)
* **Installing Tensorflow: There are a number of ways to do this.** 
  + **In Linux:**
    - **sudo pip3 install tensorflow**
  + **In Anconda Prompt (Windows):**
    - **pip install tensorflow**
* **Installing Tensorflow-GPU requires more work.**
  + **Try in Anaconda (Windows):**
    - **conda create –name tf\_gpu tensorflow-gpu**
    - **activate tf\_gpu**
    - **conda install tensorflow-gpu**
    - **conda install spyder**
    - **conda install pydot**
    - **spyder activate tf\_gpu**
  + [**ALSO**](https://www.tensorflow.org/install/gpu) **For Tensorflow-GPU: The following NVIDIA® software must be installed on your system:**
    - [NVIDIA® GPU drivers](https://www.nvidia.com/drivers)**—CUDA 10.0 requires 410.x or higher.**
    - [CUDA® Toolkit](https://developer.nvidia.com/cuda-toolkit-archive)**—TensorFlow supports CUDA 10.0 (TensorFlow >= 1.13.0)**
    - [CUPTI](http://docs.nvidia.com/cuda/cupti/) **ships with the CUDA Toolkit.**
    - [cuDNN SDK](https://developer.nvidia.com/cudnn)**(>= 7.4.1)**
  + **This** [**link**](https://towardsdatascience.com/installing-tensorflow-with-cuda-cudnn-and-gpu-support-on-windows-10-60693e46e781) **WILL help with the Tensorflow-GPU set up process by providing a number of steps with plenty of screenshots.**

**Validate in Spyder that TensorFlow and Keras work**



**Regularization, Activation Functions and Optimizers**

* **Regularization: Used to prevent overfitting. In Keras, bias, weights, and activation output can be regularized by layer.**
  + **Note: Neural networks with smaller parameter value are more insensitive to noise in the input data.**
  + **L1:**
    - **Penalty function: Favor smaller variable values using the given penalty function.**
    - **Uses a fraction of the sum of the absolute variable values.**
  + **L2** 
    - **Penalty function (same as above)**
    - **Uses a fraction of the square of the variable values**
  + **Dropout:** 
    - **Given a drop out percentage (0.45 or 45%), use the following calculation:**
      * **(1 – 0.45) \* (number of hidden units) = The remaining number of hidden units that participate for the next hidden layer.**
* **Activation Functions:** 
  + **Logistic Sigmoid: squashes weighted inputs to the range 0 to 1.**

* + **TanH: squashes weighted inputs to the range -1 to 1 (matters when negative and positive values can occur).**
  + **ReLU: squashes weighted inputs to the range 0 to the max value of x (e.g. 0 = off and max(x) = the magnitude of on).**
* **Optimizers:**
  + **Optimizers have the objective of minimizing the loss function.**
  + **Loss is the default evaluation metric in Keras, but you can choose others.**
  + **Each optimizer has tunable parameters you can select, including:**
    - **Learning Rate: Controls how much we adjust weights in our neural network relative to the loss gradient.**

### What is a Gradient?

**“A gradient measures how much the output of a function changes if you change the inputs a little bit.” — Lex Fridman (MIT)**

**Gradient descent explanation:**

* <https://towardsdatascience.com/an-introduction-to-gradient-descent-c9cca5739307>
  + - **Momentum:**
      * **Momentum is used only with the gradient to accumulate the gradient of the past steps to determine the direction to go.**
      * [**https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/**](https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/)
  + **Some example optimizers include:**
    - **Stochastic Gradient Descent (SGD):**
      * **This is one type of gradient descent. It will provide parameter updates for each set of training inputs (one after another). Which can be quicker than the original gradient descent method, but not always. It can be very expensive to compute with SGD and can create “noisy gradients” causing error to change a lot between updates.**
      * **others include:**
        + **Batch gradient descent (the original gradient descent method), calculates training error for each training example only after each training epoch completes.**
        + **Mini-Batch Gradient descent, includes methods of SGD and Batch Gradient descent to perform updates to the error by mini-batches (around “50 to 256”) instead of waiting to the end of the training epoch.**

* + - **Root Mean Squared Propagation (RMSProp): Tries to dampen oscillations during gradient descent, but not by momentum. It adjusts the learning rate automatically. A different learning rate is chosen for each parameter.**
    - **Adaptive Moments Optimization(Adam): combines momentum and RMS prop**
  + **Read this link for explanations these.**
    - * [**https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/**](https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/)