Fully connected neural network on MNIST dataset

a) Open the notebook fcn_MNIST.ipynb. In this notebook we use a fully connected neural network to predict the handwritten digits of the MNIST dataset.
We have 4000 examples with 784 pixel values and 10 classes. Run the fist 3 cells.

b) Write the missing TensorFlow code in cell 4 for the first hidden layer.

c) Run the next two cells to store the graph and do a forward pass of the untrained network. Have a look at the network.

d) Now lets train the model. Finish the code to calculate the loss and accuracy for the validation set.

Fully connected neural network on MNIST dataset (Tricks)

Note for docker users.

- In this notebook we create different runs so it might be beneficial to save them also outside the docker container. This is possible using the -v option when starting docker.

```
docker run -p 8888:8888 -p 6006:6006 -v /Users/oli/Documents/workspace/dl_course/:/notebooks
```

- If you experience crashes of the docker container do a two step procedure. First start docker in bash.

```
docker run -p 8888:8888 -p 6006:6006 -v /Users/oli/Documents/workspace/dl_course/:/notebooks
```

Then start the jupyter notebook in the console with

```
jupyter notebook --NotebookApp.token=tensorchiefs
```

a) Open the notebook fcn_MNIST_keras and run the first model (execute the cell after training) and visualize the result in TensorBoard (have a look at learning curves and the histograms / distributions of the weights)

b) Remove the `init='zero'` argument of the dense layers, to have a proper internalization of your weights. Change the name from `name = 'sigmoid_init0'` to `name = 'sigmoid'`. Restart the kernel and repeat the training as in a). Compare the results in TensorBoard, describe your results.

c) Change the activations / non-linearities from `Activation('sigmoid')` to `Activation('relu')` and change the name from `name = 'sigmoid'` to `name = 'relu'`. Continue as above, especially have a look at the validation loss do you observe overfitting.

d) Add a dropout layer: Now add a dropout layer `model.add(Dropout(0.3))` between the Dense-Layer and the Activation. Change the name from `name = 'relu'` to `name = 'dropout'`.

e) Add a batch-normalization: Now add a batch-norm layer `model.add(BatchNormalization())` between the Dense-Layer and the Dropout. Change the name from `name = 'dropout'` to `name = 'batch_dropout'`. Continue as above

The network should look like:

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
dense_1 (Dense)                  (None, 500)           392500      dense_input_1[0][0]
____________________________________________________________________________________________________
batchnormalization_1 (BatchNorma (None, 500)           2000        dense_1[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 500)           0           batchnormalization_1[0][0
____________________________________________________________________________________________________
activation_1 (Activation)        (None, 500)           0           dropout_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 50)            25050       activation_1[0][0]
____________________________________________________________________________________________________
batchnormalization_2 (BatchNorma (None, 50)            200         dense_2[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 50)            0           batchnormalization_2[0][0
____________________________________________________________________________________________________
activation_2 (Activation)        (None, 50)            0           dropout_2[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 10)            510         activation_2[0][0]
====================================================================================================
Total params: 420,260
Trainable params: 419,160
Non-trainable params: 1,100
```