Part 1:

1. **Elaborate a NN with 2 hidden fully connected layers with 300, 150 neurons and 4 neurons for classification. Use ReLU activation functions for the hidden layers and He_normal for initialization. Don't forget to flatten your image before feedforward to the first dense layer. Name the model `model_relu`.***
   Test Loss is 0.79
   Test Accuracy is 69.14 %
2. **Change the activation functions to LeakyRelu or tanh or sigmoid. Name the new model `new_a_model`. Explain how it can affect the model.***
   $tanh$ is a hyperbolic tangent function. The $tanh$ has the effect of centering the data at zero and this makes learning easier for the next layer. However it saturate for very large and small values; when we get to saturate the learning algorithm has difficulty to continue learning because the derivatives of the activation function will be zero (which mean that the weights will stop being update).
   Practically, it means that the model will tend to be saturate and will stop update when it gets data with big values.
3. **Train the new model using 25 and 40 epochs. What difference does it makes in term of performance? Remember to save the compiled model for having initialized weights for every run as we did in tutorial 12. Evaluate each trained model on the test set***
   25:
   Test Loss is 0.84
   Test Accuracy is 65.14 %
   40:
   Test Loss is 0.84
   Test Accuracy is 64.00 %
4. mini batches:
   Build the `model_relu` again and run it with a batch size of 32 instead of 64. What are the advantages of the mini-batch vs. SGD?*
   The mini-batch have low variance and better performance compared to the SGD (which mean that the SGD has low accuracy). But, it takes more time until it's converge.
   Test Loss is 0.86
   Test Accuracy is 64.57 %

   Batch normalization:

   Build the `new_a_model` again and add batch normalization layers. How does it impact your results?*

   Test Loss is 0.94

   Test Accuracy is 61.71 %

   Not as we expected the results are less good than without the batch normalization.

Part 2:

1.

- **How many layers does it have:**

24 layers.

- **How many filters in each layer?**

64,128,128,256,256.

- **Would the number of parameters be similar to a fully connected NN?**

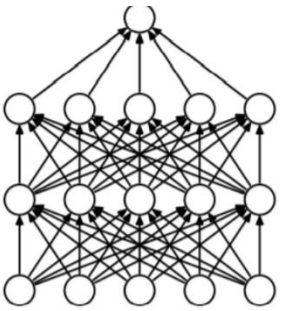
No. the whole point is that in CNN there is way less parameters then in fully connected NN.

- **Is this specific NN performing regularization?**

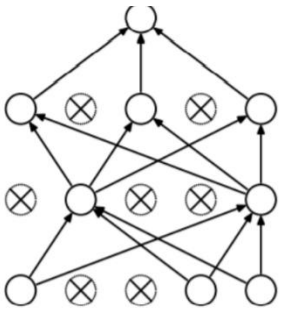
There are 2 regularization types:

(1) L2 regularization- in the syntax -"kernel_regularizer=regularizers.l2(reg))"

(2) drop-out- in the syntax - model.(Dropout(rate=dropRate))



(a) Standard Neural Net



(b) After applying dropout.

drop-out

L2 norm– ridge regression (as we learned in the previous homework).

Drop-out – Learn a fraction of the weights in the network at each training iteration. Its spread out the weights across the network in order to rely on many features.

2. **Rebuild the function get_net to have as an input argument a list of number of filters in each layers, i.e. for the CNN defined above the input should have been [64, 128, 128, 256, 256]. Now train the model with the number of filters reduced by half. What were the results.**

   Before:
   Test Loss is 7.77
   Test Accuracy is 34.29 %
   After:
   Test Loss is 5.07
   Test Accuracy is 26.29 %