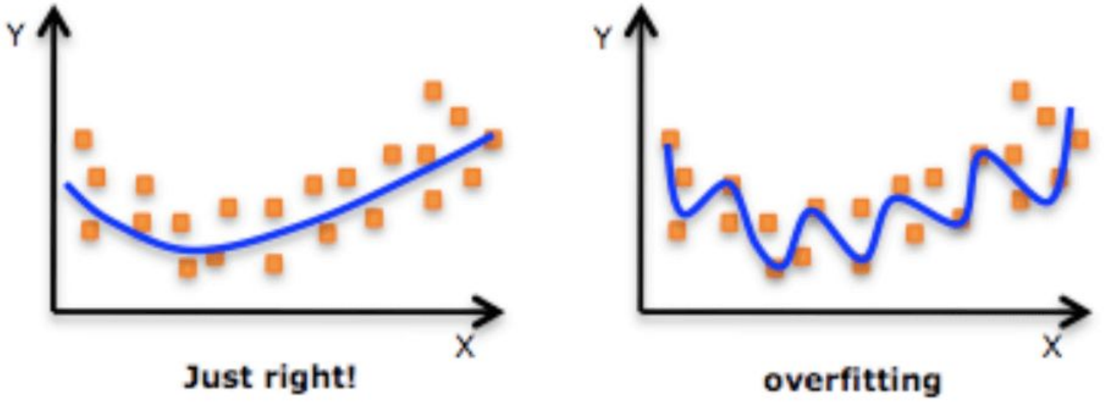
Using Dropout to Prevent Overfitting

A Project By:-

- Satwadhi Das (2016B4A70622P)
- Apurv Karpatne (2016A3PS0168P)
- Akshat Jain (2017A7PS0110P)

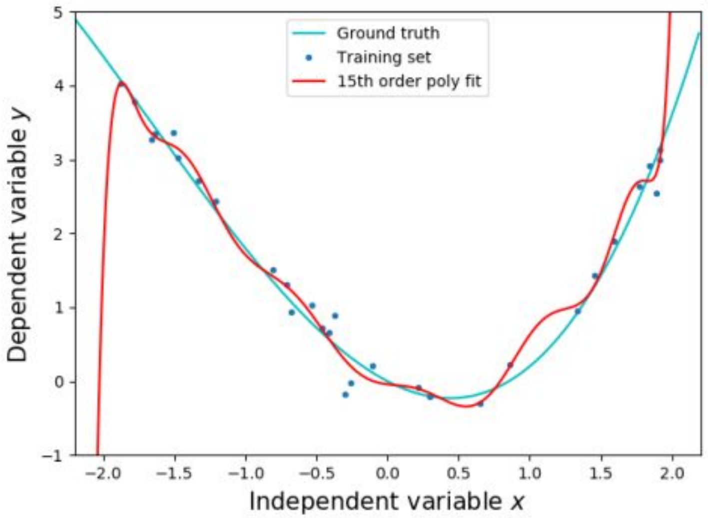
The Problem of Overfitting in Neural Networks



This image compares how an overfitted model looks with how a perfectly fit one does.

- Overfitting refers to a model that models the training data too well.
- It is a situation in which neural network is so closely fitted to the training set that it is difficult to generalize and make predictions for new data.
- It occurs when a model tries to predict a trend in data that is too noisy. This is the caused due to an overly complex model with too many parameters.

Why is Overfitting a Problem?

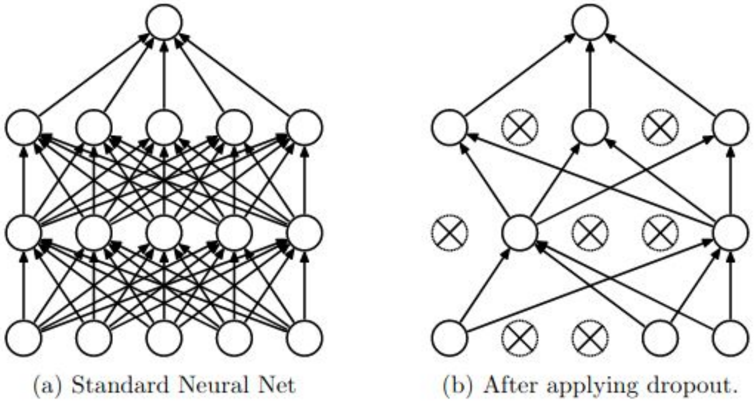


This image shows a model overfitted to account for the observed noise in the training set, creating artificial waviness.

Overfitting is bad because:

- The model has extra capacity to learn the random noise in the observation
- To accommodate noise, an overfit model overstretches itself and ignores domains not covered by data.
- Consequently, the model makes poor predictions everywhere other than near the training set. In other words, it does not generalize well, and can't be used to make prediction for new data.

What is Dropout and how Does it Help?

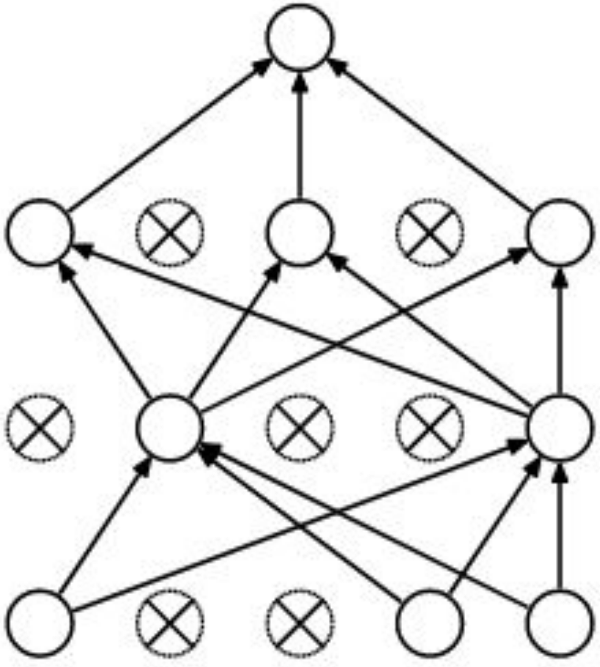


Dropout Neural Net Model.
Left: A standard neural net with 2 hidden layers.
Right:An example of a thinned net produced by applying dropout to the network on the left.

Dropout refers to dropping out units (hidden and visible) in a neural network. When we drop different sets of neurons, it's equivalent to training different neural networks. The different networks will overfit in different ways, so the net effect of dropout will be to reduce overfitting.

Dropout v/s other regularization

- Extremely useful in places with limited data sets.
- Essentially, 2^n networks with shared weights can be combined into a single neural network to be used at test time
- Complex co-adaptations can be trained to work well on a training set, but on novel test data they are far more likely to fail than multiple simpler co-adaptations that achieve the same thing. This is the exact principle behind dropout



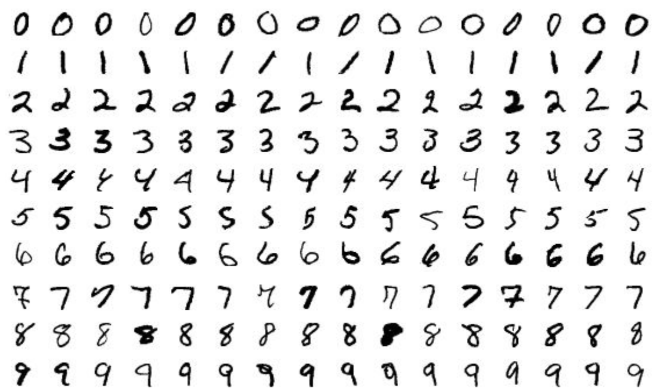
Comparison of different models on MNIST

Method	Unit Type	Architecture	Error %
Standard Neural Net (Simard et al., 2003)	Logistic	2 layers, 800 units	1.60
SVM Gaussian kernel	NA	NA	1.40
Dropout NN	Logistic	3 layers, 1024 units	1.35
Dropout NN	ReLU	3 layers, 1024 units	1.25
Dropout NN + max-norm constraint	ReLU	3 layers, 1024 units	1.06
Dropout NN + max-norm constraint	ReLU	3 layers, 2048 units	1.04
Dropout NN + max-norm constraint	ReLU	2 layers, 4096 units	1.01
Dropout NN + max-norm constraint	ReLU	2 layers, 8192 units	0.95
Dropout NN + max-norm constraint (Goodfellow et al., 2013)	Maxout	2 layers, (5×240) units	0.94
DBN + finetuning (Hinton and Salakhutdinov, 2006)	Logistic	500-500-2000	1.18
DBM + finetuning (Salakhutdinov and Hinton, 2009)	Logistic	500-500-2000	0.96
DBN + dropout finetuning	Logistic	500-500-2000	0.92
DBM + dropout finetuning	Logistic	500-500-2000	**0.79**

Table 2: Comparison of different models on MNIST.

The Project’s Data Specifics

- For the project, MNIST dataset was used.
- It is a set of 28 x 28 pixel images containing handwritten number digits.
- It contains 60,000 train samples and 10,000 test samples.
- MNIST is used because it is decently large, but not so large that it becomes unwieldy to train our network.
- It allows to train robust neural networks.



MNIST examples

Processing Done on Data

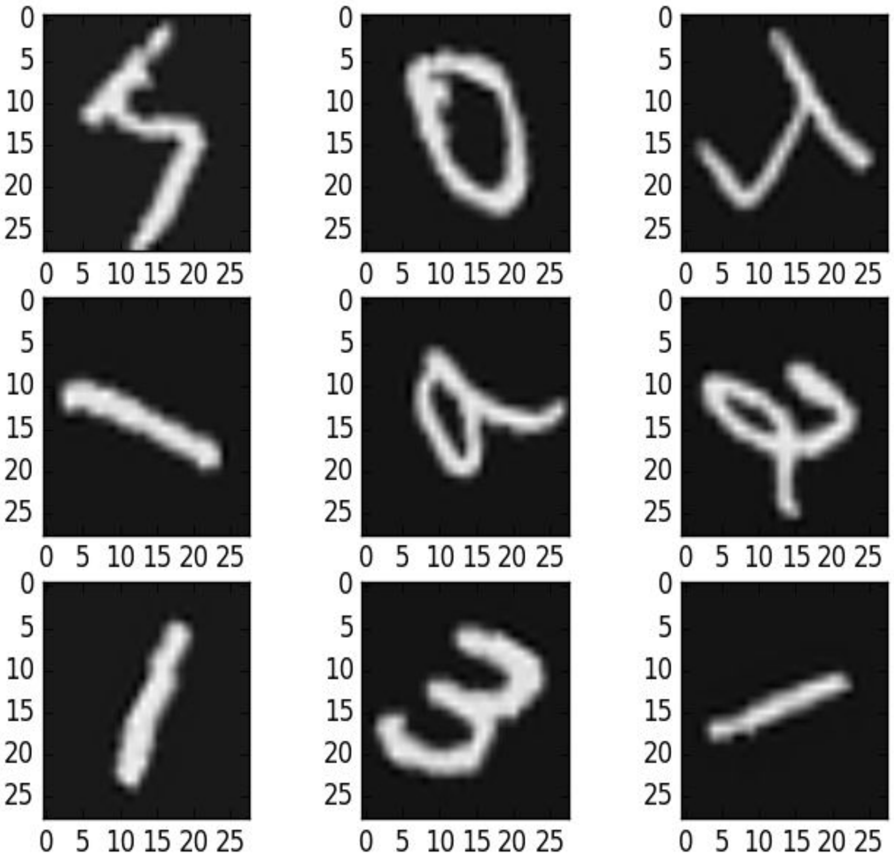
- Re-Shaping and Normalising the Data
    - To be able to use the dataset in Keras API, we need 4-dims numpy arrays.
    - We must normalize our data as it is always required in neural network models. We can achieve this by dividing the RGB codes to 255 (which is the maximum RGB code minus the minimum RGB code)

Data Augmentation

- Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data

- Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks.

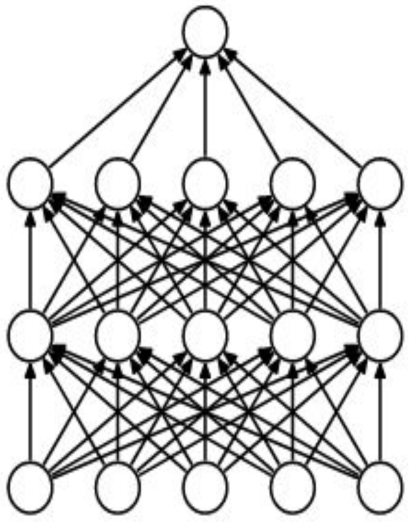
- We use ImageDataGenerator to implement Data Augmentation in our model



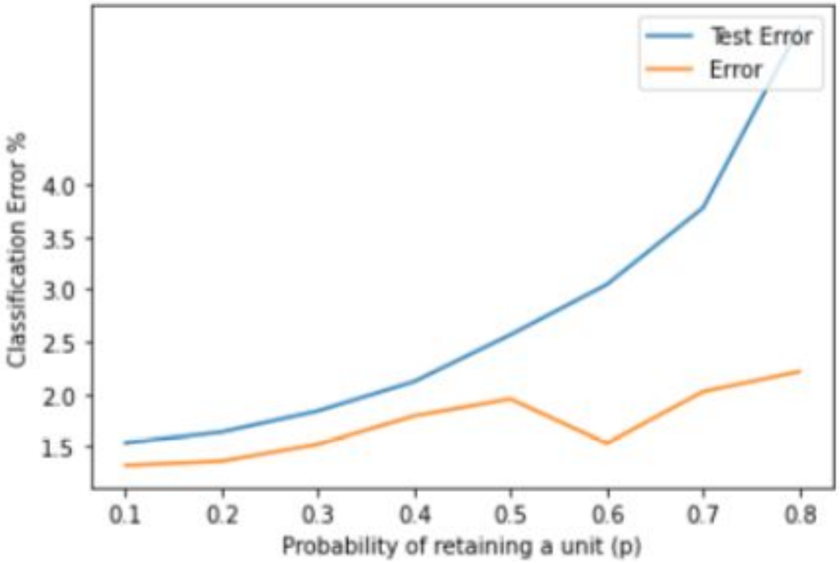
Result of Data Augmentation

The Model

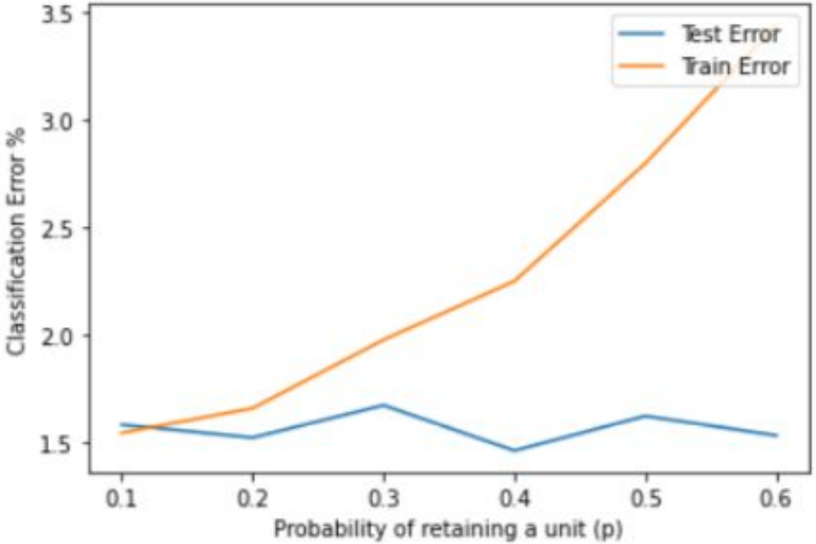
- The network architecture used is (784-1024-1024-2048-10)
- ReLU activation is used with stochastic gradient descent
- The network is trained using two different techniques-
  - Keeping the architecture constant (n)
  - Keeping the number of nodes in each iteration constant (pn)
- For each technique, 9 probabilities [0.1, 0.2,...0.9] were trained



Findings from Our Project



Keeping ‘n’ constant



Keeping ‘pn’ constant

Discussion of Results-

- A simple Neural Network gave a test error of 1.59%.
- The neural network used for this purpose had a 784-128-128-128-10 architecture
- Our NN with dropout had a better performance than a simple neural network and was in-line with the research paper's finding
- Our models had a flat PSNR of around -14.491539

Conclusions and limitations-

- Our models achieved desirable results, the performance of our Neural Network was better than one without dropout.
- Implementing and using Data Augmentation made the neural network more robust
- One major limitation was the amount of time required to train the models. This prevented fine tuning the model even more to achieve better results.

Thank You.