Final Project

DATS 6203

Spring 2018

Group 6

Vinay Bhandaru, Joseph Haaga

Introduction:

The goal of this project is to train a convolutional neural network to classify handwritten digits based on the SVHN dataset. This report will describe the dataset, network architecture, how the network will be trained, and then how the trained model performs on new data.

Description of the dataset:

The Street View House Numbers (SVHN) Dataset consists of images of house numbers in Google Street View images. Each image is classified as 1 of 10 digits from 0 to 9. There are 73257 digits for training and 26032 digits for testing. Although the original images with bounding boxes around individual characters is available, the images that will be used for this project are cropped images of single characters resized to a resolution of 32-by-32 pixels. The dimension of each images is 32X32X3. 3 represents values of red, green, and blue color channels in a color image.

Description of the deep learning network and training algorithm:

The deep learning network that will be used will be composed of 1 or more 2-dimensional convolutional layers each followed by a max pooling layer and then the rectified linear unit activation function. Then there will be a fully connected layer consisting of hundreds of neurons, followed by a softmax layer of 10 neurons. A dropout layer prior to the fully connected layer will also be implemented. An example of a CNN architecture that will be used is shown below.

CNN architecture
Conv2d(3, 20, kernel_size=(5, 5), stride=(1, 1))
MaxPool2d (size=(2, 2), stride=(2, 2), dilation=(1, 1))
ReLU ()
Conv2d(20, 50, kernel_size=(5, 5), stride=(1, 1))
MaxPool2d (size=(2, 2), stride=(2, 2), dilation=(1, 1))
ReLU ()
Dropout2d (p=0.25)
Flatten ()
Linear (1250 -> 500)
ReLU ()
Linear (500 -> 10)
LogSoftmax ()

The training algorithm will be either stochastic gradient descent or the Adam algorithm with mini-batches. The framework that will be used to implement CNN networks and training algorithms is the Pytorch module in Python utilizing GPU.

Experimental Setup:

Each CNN network will be trained on the 73,257 training images in pytorch utlizing GPU. Then each trained model will be evaluated on the 26,032 test images. The goal will be to find the best network in terms of classification accuracy. The primary metric will be overall accuracy and accuracy for each digit.

Mini-batches will be used to decrease the computation time. To determine the optimal number of mini-batches, the computation time for stochastic gradient descent with 2 epochs based on different mini-batch sizes and 2 convolution layers will be compared, and the value with the lowest computation time will be selected.

Optimization algorithms that will be compared are stochastic gradient descent and Adam, each with several learning rates. 7000 samples from the training set will be reserved for validation and the remaining 66257 for training. In order to determine which algorithm and learning rate to use, the accuracy in the validation set will be plotted over the number of epochs for each method based on a network with 2 convolution layers.

During training of the network, over-fitting can be detected by comparing the accuracy on the validation or test set to the accuracy on the training set. For example, if the training accuracy is substantially higher, this indicates over-fitting and that a model with fewer parameters might be preferable.

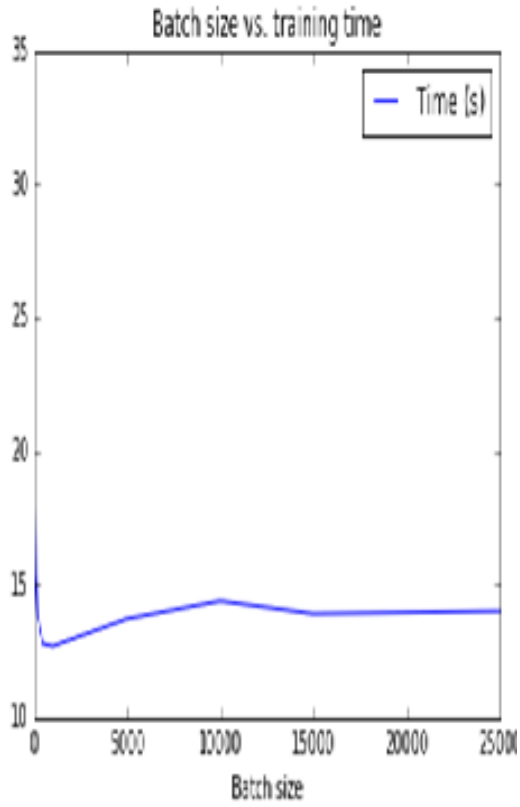
In addition, normalizing the images in each of the 3 channels will be attempted to determine if there is a benefit.

Once the optimal pre-processing and training parameters are selected, several CNNS with different numbers of layers will be compared to determine if the number of convolutional layers makes a difference. This will be evaluated by adding convolutional layers with 20 kernals of size 5 and stride 1 followed by a max pooling layer of size 2 with stride 1. The number of neurons in the fully connected layer will be adjusted so that each model has roughly 1.4 million parameters to train. The accuracy on the test set will be compared.

Results:

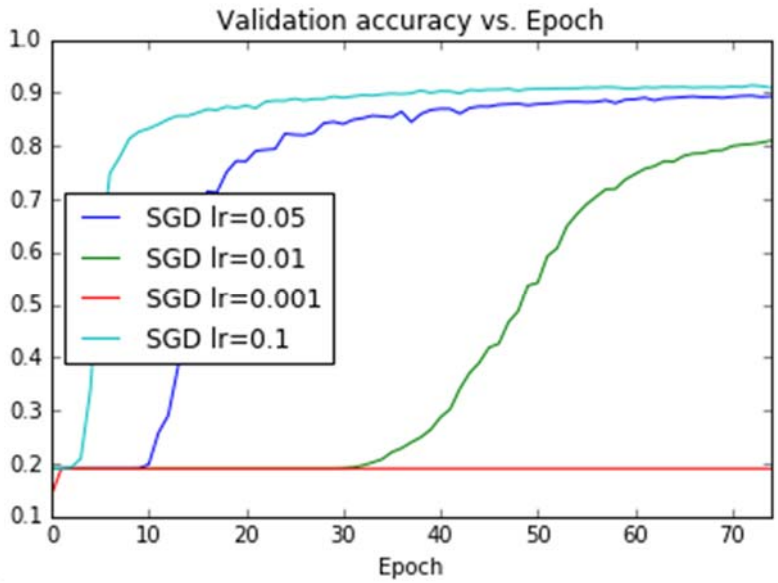
Batch Size:

Batch size	Time (s)
16	32.1282
50	18.3828
100	15.3778
200	13.7388
500	12.7581
1000	12.6773
5000	13.6990
10000	14.3668
15000	13.8882
25000	13.9889



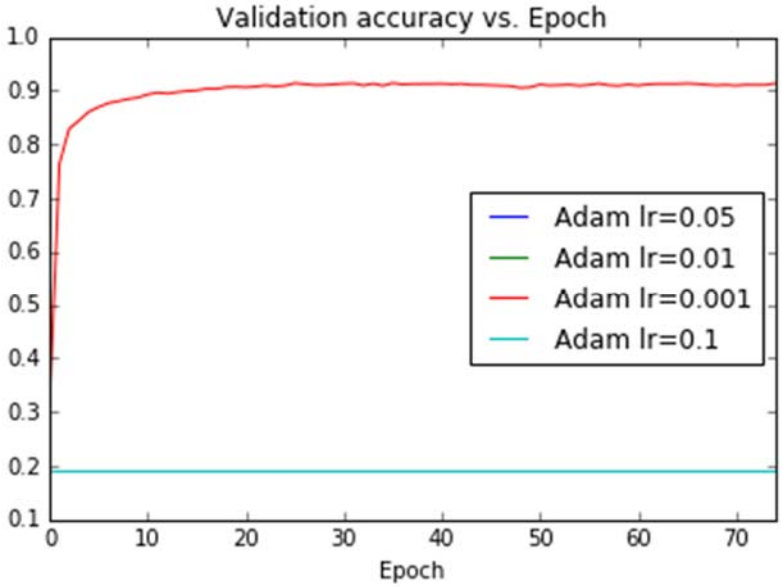
The computation time for various batch sizes is displayed in the above figure. The times are based on 2 epochs of training a convolution network with 2 convolution layers and 1 fully connected layer with 500 neurons. A mini-batch size of 1000 resulted in the smallest computation time, so that is the selected value.

Learning rate and optimization algorithm

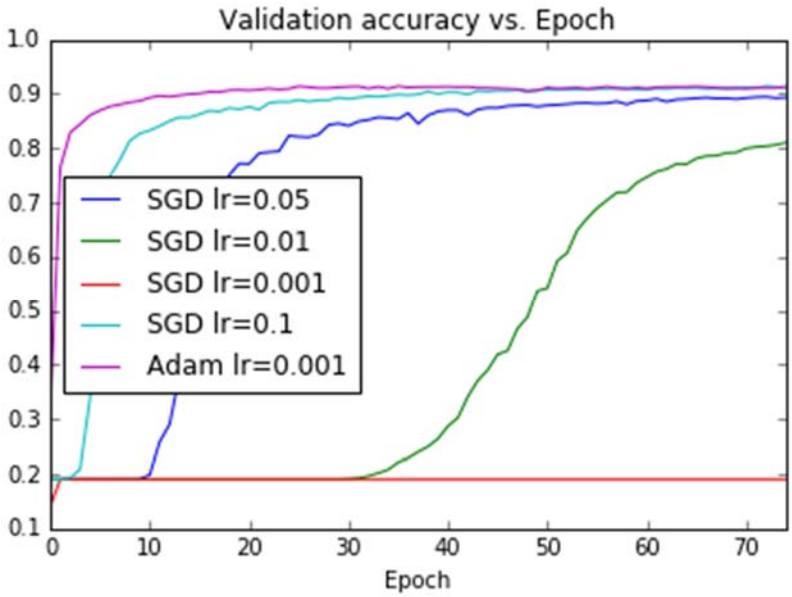


/

The above figure plots the accuracy in the validation set for each epoch based on stochastic gradient descent optimization with various learning rates. The best learning rate is 0.1 which has the highest validation accuracy at every epoch. The next best is a learning rate of 0.05 followed by 0.01. The red curve is for a learning rate of 0.001 and there was no increase in accuracy unlike the other 3 curves.



The above figure plots the accuracy in the validation set for each epoch based on Adam optimization with various learning rates. The best learning rate is 0.001 which has the highest validation accuracy at every epoch. For all other learning rates there was no increase in accuracy.



The above figure plots the results of Adam with a learning rate of 0.001 with the results of stochastic gradient descent. Adam achieves a higher validation accuracy faster, but eventually SGD with a learning rate of 0.1 appears to converge. Based on these results, Adam with a learning rate of 0.001 is preferable.

CNN training and prediction results

The table below displays various measures for the convolutional neural network with 2 convolutional layers, a 2-dimensinal dropout layer, a fully connected dense layer consisting of 500 neurons and the rectified linear unit activation function followed by a second fully connected layer of 10 neurons and the log softmax activation function.

Digit	*Precision*	*Accuracy*	*F1-score*	*Support*
0	0.89	0.91	0.90	1744
1	0.92	0.94	0.93	5099
2	0.94	0.92	0.93	4149
3	0.87	0.84	0.85	2882
4	0.88	0.93	0.90	2523
5	0.91	0.90	0.90	2384
6	0.88	0.88	0.88	1977
7	0.92	0.89	0.91	2019
8	0.90	0.83	0.86	1660

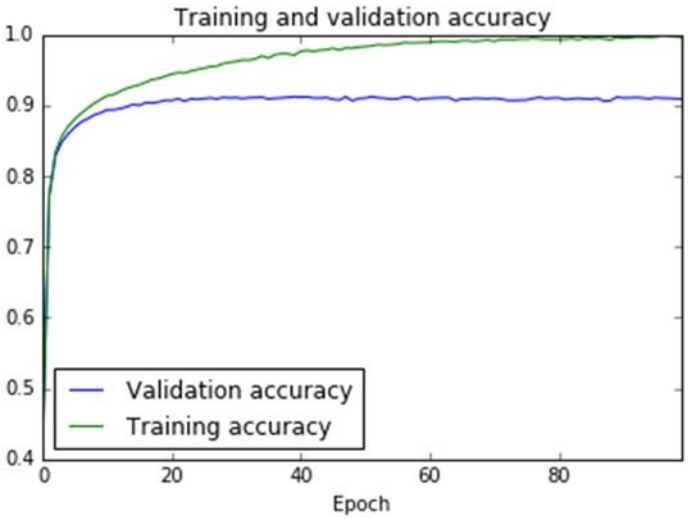
Digit	*Precision*	*Accuracy*	*F1-score*	*Support*
9	0.79	0.89	0.84	1595
Avg/Total	0.90	0.90	0.90	26032

The overall accuracy of the CNN model is 90%. Images of 1 were classified most accurately (94%) while images of 3 and 8 were classified least accurately at 84% and 83% respectively.

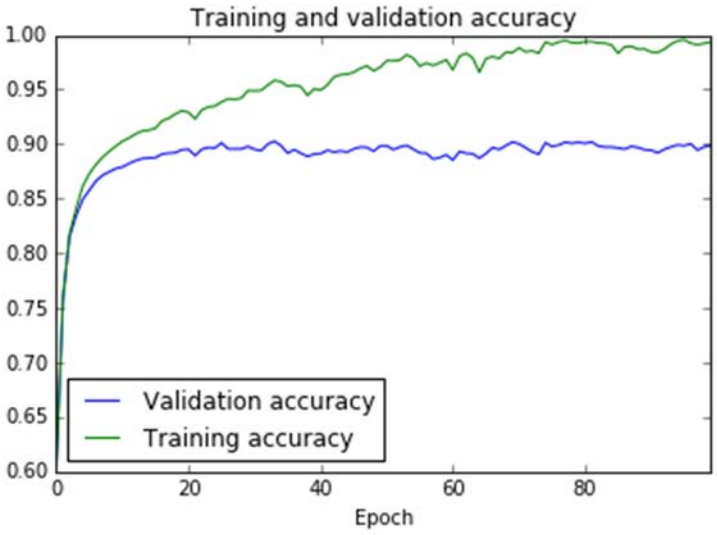
	Predicted									
Actual	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	1590	22	10	19	9	7	38	5	11	33
1	43	4780	47	32	105	11	15	47	7	12
2	14	50	3823	70	58	15	17	53	21	28
3	18	95	43	2407	25	77	20	12	36	149
4	16	60	29	19	2334	6	10	15	11	23
5	7	15	19	84	25	2135	54	4	7	34
6	41	16	11	27	33	44	1747	5	31	22
7	5	118	41	31	16	3	5	1791	2	7
8	22	13	10	51	29	27	73	4	1373	58
9	28	18	40	21	19	18	11	3	20	1417

The cross-tabulation above shows how images of each digit were classified in the test set. The largest numbers are across the diagonal, indicating correct classification. For example, images of the number 3 were mostly classified as 3, but also misclassified as 9, 1, or 5, and the number 8 was misclassified as 9 or 3.

There is some evidence of over-fitting in this model indicated by the plot below. The training accuracy increases to 1, but the validation accuracy increases to 0.9 and does not increase beyond that. Therefore, a model with fewer parameters might be preferable.



Removing the dropout layer results in a slight decrease in accuracy to 0.89. In addition, the training and validation accuracies fluctuate more as indicated by the plot below. Therefore, we will assume the dropout layer is important.



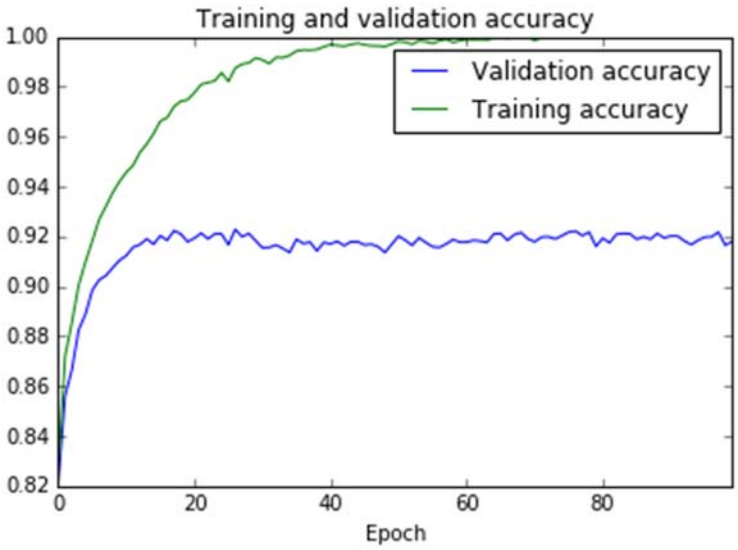
Normalization of the images was performed based on the means and standard deviations in the training dataset in each of the 3 channels. The resulting model accuracy increased from 89.9% to 91.2% as shown in the table below.

With normalization

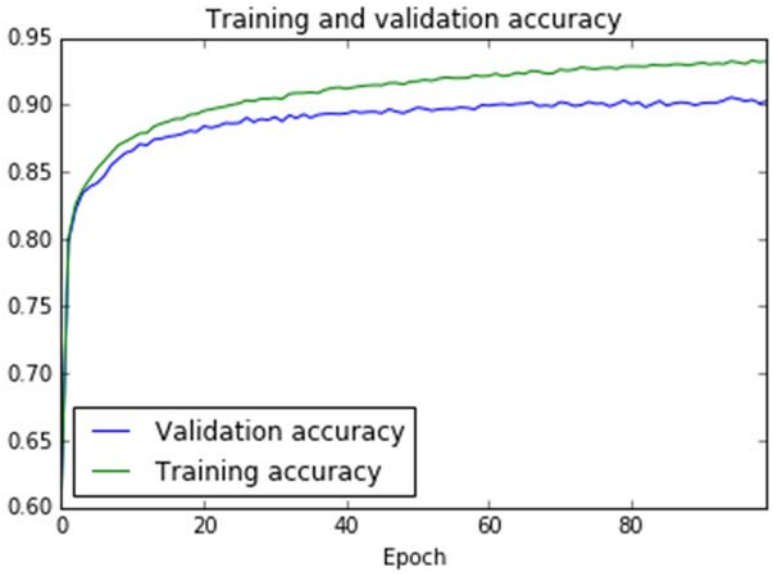
Digit	*Precision*	*Accuracy*	*F1-score*	*Support*
0	0.91	0.91	0.91	1744
1	0.94	0.94	0.94	5099
2	0.91	0.95	0.93	4149
3	0.89	0.86	0.87	2882
4	0.93	0.91	0.92	2523
5	0.92	0.91	0.92	2384
6	0.87	0.89	0.88	1977
7	0.94	0.91	0.92	2019
8	0.86	0.88	0.87	1660
9	0.87	0.87	0.87	1595
Avg/Total	0.91	0.91	0.91	26032

The confusion matrix is shown below.

					Predicted					
Actual	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	1579	19	22	14	3	8	46	9	22	22
1	47	4787	57	53	59	10	18	44	18	6
2	6	27	3953	48	30	11	17	34	12	11
3	11	66	64	2480	14	61	25	14	55	92
4	13	58	45	25	2304	12	11	10	20	25
5	6	14	22	69	10	2177	51	5	20	10
6	26	18	18	32	13	37	1763	5	54	11
7	2	72	56	17	5	10	10	1839	4	4
8	15	16	29	30	14	13	62	2	1460	19
9	33	12	56	28	13	21	12	2	23	1395



The above plot indicates that training accuracy was 100% after 65 epoch, while validation accuracy was around 92%. This again indicates overfitting. A second model was fit with only 20 kernals in the second convolution layer and 300 neurons instead of 500 in the fully connected layer. This is a reduction from 657,080 parameters to 164,850 parameters. However, the accuracy of this model on the test set was only 90.1%, indicating that it's not preferable. Reducing the second convolution layer from 50 to 10 kernals and the fully connected layer to 50 neurons results in a model with 19,590 parameters. The test accuracy is however, 89.8%. There is less overfitting with this model as indicated in the plot below, but the accuracy is substantially less. Therefore, we'll assume that the model with the most parameters is best.



The training time for 100 epochs for the model with the most parameters was 5 minutes and 11 seconds. The two models with fewer parameters trained in around 4 minutes and 25 seconds.

Comparing training time and accuracy based on layers

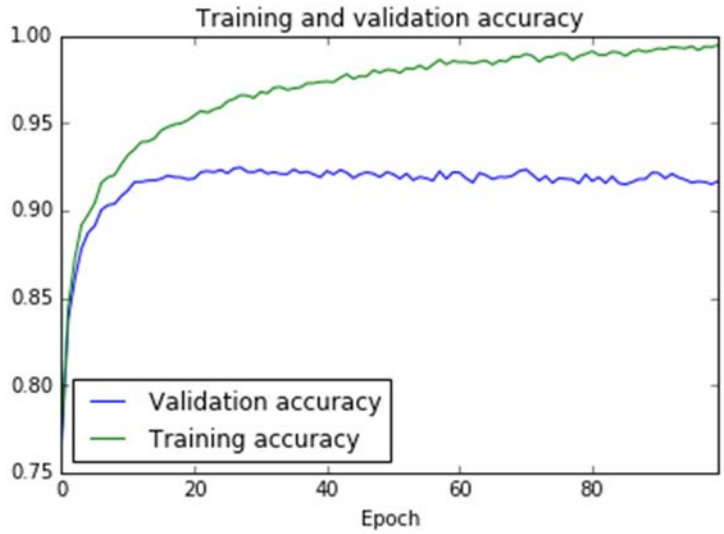
The results of the comparison of models with different numbers of layers are shown in the table below.

# *layers*	*Time (s)*	*Accuracy*
1	71.077	0.83340
2	375.280	0.88303
3	453.275	0.90565
4	508.127	0.89705
5	538.272	0.89866

The above table is based on convolutional layers with 20 kernals of size 5 and stride 1 followed by a max pooling layer of size 2 with stride 1. The number of neurons in the fully connected layer was adjusted so that each model has roughly 1.4 million parameters to train. Training time increases with number of layers, but accuracy is highest for 3 layers.

Best CNN model:

Motivated by the above results, an alternative 3-layer model with 656,830 parameters similar to the best-performing 2-layer but with an additional convolutional layer with 50 kernels and 2 max pooling layers with stride 1 instead of 2 was trained and had the best performance (91.6%) on the test set. The training and validation accuracy are plotted below as is the prediction accuracy on the test set.



Digit	*Precision*	*Accuracy*	*F1-score*	*Support*
0	0.93	0.89	0.91	1744
1	0.94	0.95	0.94	5099
2	0.93	0.95	0.94	4149
3	0.87	0.90	0.88	2882
4	0.95	0.91	0.93	2523
5	0.93	0.90	0.91	2384
6	0.89	0.90	0.89	1977
7	0.93	0.90	0.92	2019
8	0.87	0.89	0.88	1660
9	0.89	0.90	0.89	1595
Avg/Total	0.92	0.92	0.92	26032

The confusion matrix is

	Predicted									
Actual	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	1557	21	17	22	6	7	46	10	17	41
1	27	4823	43	59	38	18	9	46	32	4
2	5	30	3932	72	14	15	13	32	26	10
3	9	43	48	2599	9	54	15	14	39	52
4	9	85	29	32	2293	12	18	13	16	16
5	7	13	17	113	14	2137	48	4	17	14
6	20	14	16	29	10	45	1778	4	49	12
7	5	99	50	24	5	2	2	1820	6	6
8	13	13	13	29	12	14	57	4	1476	29
9	23	16	52	17	9	6	11	5	27	1429

Images of 1 were classified most accurately (95%) while images of 0 and 8 were classified least accurately at 89%. The cross-tabulation above shows how images of each digit were classified in the test set. Both 0 and 8 were most often misclassified as 6.

Summary and Conclusions

We determined that the optimal way to train a CNN on the SVHN data with Pytorch and GPU is with Adam and a learning rate of 0.001 and a mini-batch size of 1000. In addition, normalization is important. The plots indicate that there is overfitting in all of the CNNs that were trained. However, reducing the number of parameters resulted in a slight decrease in prediction accuracy with an insignificant gain in computational speed. The best CNN model has 3 convolution layers and 656,830 parameters and is able to classify 91.6% of the test images correctly.

Additional parameters that might result in improved performance include padding in the convolution layer.

References

Diederik P. Kingma, Jimmy Ba Adam: A Method for Stochastic Optimization. 3rd International Conference for Learning Representations, San Diego, 2015

http://ufldl.stanford.edu/housenumbers/

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

https://codetolight.wordpress.com/2017/11/30/getting-started-with-pytorch-for-deep-learning-part-3-5-pytorch-sequential/

pytorch.org