Report

Srinidhi Goud

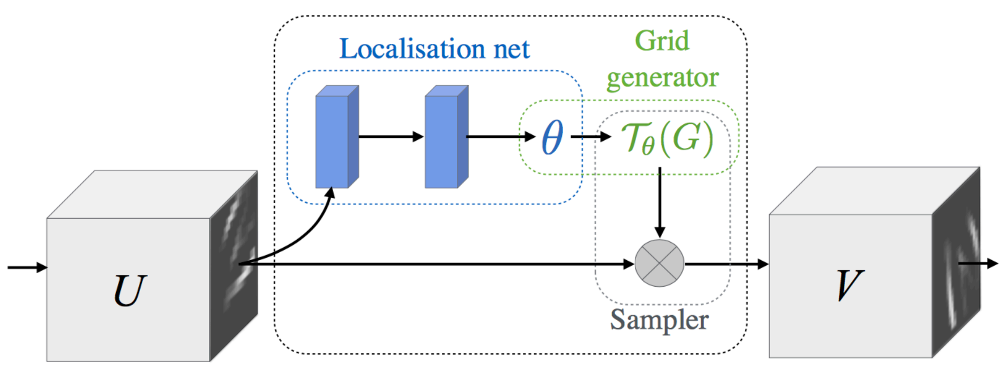
This project was completed with over 99% accuracy model, but due to some problem my csv file was giving gibberish results even though it shows over 99% accuracy on validation set. I have mailed Professor and Rohit regarding this. Even now I cannot submit late as I cannot make the output file right! Hope this would be considered while evaluating my code. Please look at my slurm command, I am attaching a model as well, to run to test the accuracy.

Models implemented in increasing order of performance:
Traffic Sign Recognition with Multi-Scale Convolutional Networks {sermant, yann}(98.4)
Multi-Column Deep Neural Network for Traffic Sign Classification {Dan Ciresan et al}
Traffic Sign Recognition with Hinge Loss Trained Convolutional Neural Networks {Junqi Jin et al}(99%+)

Data Preprocessing:

As the given data might be too uniform and so, to prevent the model to over fit on the training data and get better accuracy, I have used data augmentation using Transforms, and imported another module called 'torchsample' as Transform alone cannot do affine transformation on the images. The source for writing this torch sample is: ***Nick Collen (ncullen93) from GitHub.***

Then I have Spatial Transformer Network to learn parameters to transform images spatially invariant before feeding it to the network. This caused a high jump in accuracy. Reference for **Spatial Transformer Network is this:**
Spatial Transformer Networks {Max Jaderberg et al }



Spatial Transformer

Moreover, to remove the data biased due, input data imbalances, I have sample lower frequency data, using Weight Sampler in PyTorch. This on Jin's model gave me the highest accuracy of over 99% (its over 99% as in pytorch I am outputting integer, but in kaggle traditionally the score increases to some extent compared to the noted accuracy, locally)

However, even though Data augmentation works on Yann's model well enough, it is very unstable on Spatial Transformer Network, causing it to fall down to a low value and remain staying there. To avoid this Data augmentation has been removed, learning rate has been reduced and Batch size has also been reduced. This way, even though I have used 80 epochs, model reaches 99% accuracy in 37 epochs.

Model Architectures:

Yann's model:

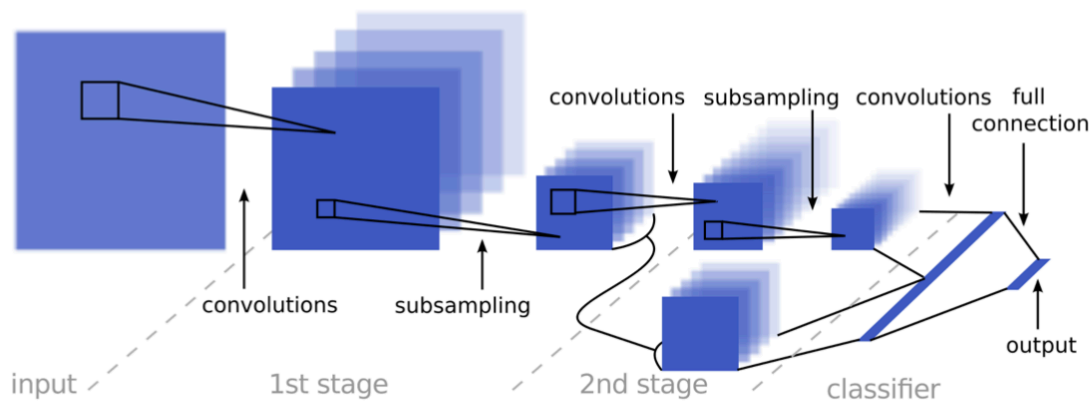


Fig. 2. A 2-stage ConvNet architecture. The input is processed in a feed-forward manner through two stage of convolutions and subsampling, and finally classified with a linear classifier. The output of the 1st stage is also fed directly to the classifier as higher-resolution features.

Note: Commented out part in my submitted file shows how I aoncat output of stage 1 to output of stage 2 after sub sampling.

Jin's model:

JIN *et al.*: TRAFFIC SIGN RECOGNITION WITH HINGE LOSS TRAINED CONVOLUTIONAL NEURAL NETWORKS 1995

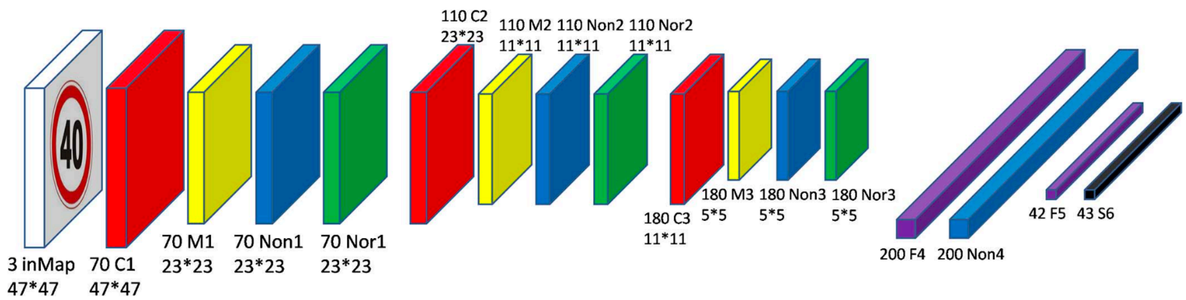


Fig. 4. Our CNN's architecture: inMap → C1 → M1 → Non1 → Nor1 → C2 → M2 → Non2 → Nor2 → C3 → M3 → Non3 → Nor3 → F4 → Non4 → F5 → S6.

As I have used 32x32 scaled images, conv1(3x70), conv2(70x110), conv3(110x180), Fc1(4*4*180, 200), Fc2(200, 43)

Given low number of parameters that give 99% accuracy this model is the best.

I have used Relu as activation function after each layer, Max pooled with stride two for subsampling, used Batch Normalization to regularize the weights. This way we ensure my model converges in fastest time.

Result: after epoch 38

Train Epoch: 38 [0/18715 (0%)] Loss: 0.007428
Train Epoch: 38 [320/18715 (2%)] Loss: 0.000804
Train Epoch: 38 [640/18715 (3%)] Loss: 0.012250
Train Epoch: 38 [960/18715 (5%)] Loss: 0.026386
Train Epoch: 38 [1280/18715 (7%)] Loss: 0.002512
Train Epoch: 38 [1600/18715 (9%)] Loss: 0.006526
Train Epoch: 38 [1920/18715 (10%)] Loss: 0.014469
Train Epoch: 38 [2240/18715 (12%)] Loss: 0.157243
Train Epoch: 38 [2560/18715 (14%)] Loss: 0.014243
Train Epoch: 38 [2880/18715 (15%)] Loss: 0.029755
Train Epoch: 38 [3200/18715 (17%)] Loss: 0.017645
Train Epoch: 38 [3520/18715 (19%)] Loss: 0.003040
Train Epoch: 38 [3840/18715 (21%)] Loss: 0.003460
Train Epoch: 38 [4160/18715 (22%)] Loss: 0.002557
Train Epoch: 38 [4480/18715 (24%)] Loss: 0.026290
Train Epoch: 38 [4800/18715 (26%)] Loss: 0.171526
Train Epoch: 38 [5120/18715 (27%)] Loss: 0.008663
Train Epoch: 38 [5440/18715 (29%)] Loss: 0.050791
Train Epoch: 38 [5760/18715 (31%)] Loss: 0.047584
Train Epoch: 38 [6080/18715 (32%)] Loss: 0.032809
Train Epoch: 38 [6400/18715 (34%)] Loss: 0.004364

Train Epoch: 38 [6720/18715 (36%)] Loss: 0.035160
Train Epoch: 38 [7040/18715 (38%)] Loss: 0.009796
Train Epoch: 38 [7360/18715 (39%)] Loss: 0.042059
Train Epoch: 38 [7680/18715 (41%)] Loss: 0.004183
Train Epoch: 38 [8000/18715 (43%)] Loss: 0.059701
Train Epoch: 38 [8320/18715 (44%)] Loss: 0.008306
Train Epoch: 38 [8640/18715 (46%)] Loss: 0.007344
Train Epoch: 38 [8960/18715 (48%)] Loss: 0.018173
Train Epoch: 38 [9280/18715 (50%)] Loss: 0.007012
Train Epoch: 38 [9600/18715 (51%)] Loss: 0.005434
Train Epoch: 38 [9920/18715 (53%)] Loss: 0.020878
Train Epoch: 38 [10240/18715 (55%)] Loss: 0.005528
Train Epoch: 38 [10560/18715 (56%)] Loss: 0.071100
Train Epoch: 38 [10880/18715 (58%)] Loss: 0.002309
Train Epoch: 38 [11200/18715 (60%)] Loss: 0.006241
Train Epoch: 38 [11520/18715 (62%)] Loss: 0.011361
Train Epoch: 38 [11840/18715 (63%)] Loss: 0.005081
Train Epoch: 38 [12160/18715 (65%)] Loss: 0.003862
Train Epoch: 38 [12480/18715 (67%)] Loss: 0.007520
Train Epoch: 38 [12800/18715 (68%)] Loss: 0.023978
Train Epoch: 38 [13120/18715 (70%)] Loss: 0.009998
Train Epoch: 38 [13440/18715 (72%)] Loss: 0.015948
Train Epoch: 38 [13760/18715 (74%)] Loss: 0.003670
Train Epoch: 38 [14080/18715 (75%)] Loss: 0.009961
Train Epoch: 38 [14400/18715 (77%)] Loss: 0.004007
Train Epoch: 38 [14720/18715 (79%)] Loss: 0.011443
Train Epoch: 38 [15040/18715 (80%)] Loss: 0.003544
Train Epoch: 38 [15360/18715 (82%)] Loss: 0.002042
Train Epoch: 38 [15680/18715 (84%)] Loss: 0.042416
Train Epoch: 38 [16000/18715 (85%)] Loss: 0.006781
Train Epoch: 38 [16320/18715 (87%)] Loss: 0.017557
Train Epoch: 38 [16640/18715 (89%)] Loss: 0.032024
Train Epoch: 38 [16960/18715 (91%)] Loss: 0.004567
Train Epoch: 38 [17280/18715 (92%)] Loss: 0.013028
Train Epoch: 38 [17600/18715 (94%)] Loss: 0.001854
Train Epoch: 38 [17920/18715 (96%)] Loss: 0.024910
Train Epoch: 38 [18240/18715 (97%)] Loss: 0.008254
Train Epoch: 38 [18560/18715 (99%)] Loss: 0.010791

Validation set: Average loss: 0.0455, Accuracy: 2080/2109 (99%)