Classification of Image Data

Donghan Liu, Dingtao Hu, Jifeng Wang
McGill University COMP551 Mini project3

***Abstract*— In this project, we were tasked to develop two neural networks models Multilayer Perceptrons (MLP) and Convolutional Neural Networks (CNN) to implement classification on image data from CIFAR10 data set to recognize the class of the picture. We improved the performance of MLP by hyperparameter-tuning and architecture adjustment and modified the CNN model from the tutorial by adding convolutional layers and changing the dropout.CNNs archived around 70% accuracy while the best MLPs were at the level of 51%. Furthermore, CNN reached high accuracy when identifying artificial objects. Additionally, both two models took 3-4 hours for training which is extremely time-consuming. To further optimize the better-performance model on the training time, we increased the learning rate and batch size for CNN based on the theory of Super-Convergence which archived a faster training in our experiment.**

I. Introduction

The project focuses on the following tasks: implementing two artificial neural networks models Multilayer Perceptrons (MLP) and Convolutional Neural Networks (CNN) to classify the image data from CIFAR10 data set; designing mini-batch gradient descent for the backpropagation process; improving the performance of MLP by hyperparameter-tuning and architecture adjustment on the number of layers, batch size and activation function; modifying CNN by adding at most two convolutional layers with batch normalization, changing the dropout and batch size; optimizing CNN's training time by increasing the learning rate and batch size to converge faster to some level of loss.[10]

This experiment applied the CIFAR10 dataset. It has the classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. The images in CIFAR-10 are of size 3x32x32, i.e. 3-channel color images of 32x32 pixels in size. A package called torchvision, was used as the data loader for CIFAR10 and data transformer for images and torchvision.datasets. The output of torchvision datasets are PILImage images of range [0, 1]. Tey were transformed to Tensors of normalized range [-1, 1] by torchvision.transforms. Additionally, we standardized image data by Normalizer from sklearn.preprocessing to speed up training and archive higher accuracy/lower loss. The CNN after modification and adjustment archived 0.69 accuracy while the best MLP group among all 11 experimental groups still remained at the level of 0.51, one of the worst one even touched the bottom down to 0.11. The setting of a larger learning rate for CNN converged fast within tens of epoch. The further research will focus on the optimal extent of improvement to learning rate.

II. Related Work

A common use of machine learning is to identify what an image represents. The task of predicting what an image represents is called image classification. An image classification model is trained to recognize various classes of images by the probabilities of the image representing each of the class.[1] For example, a model might be trained to recognize photos representing three different types of animals: rabbits, hamsters, and dogs.

In this project, we mainly focused on the application of neural networks on the image classification.Generally, deep learning is the application of artificial neural networks using modern hardware.It allows the development, training, and use of neural networks that are much larger (more layers) than was previously thought possible.There are thousands of types of specific neural networks proposed by researchers as modifications or tweaks to existing models.Sometimes wholly new approaches. Among all artificial neural networks, Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs)provide a lot of flexibility and have proven themselves over decades to be useful and reliable in a wide range of problems.[2]

The findings from the reference[3] indicate that the CNN image classification method for CIFAR-10 has achieved a higher accuracy (0.62) in comparison with MLP (0.44), and another reference[4] indicates a convolutional neural network that can recognize images from CIFAR-10 data set with an accuracy of 78% using TensorFlow.

III. Dataset and Setup

A. Dataset Description

The CIFAR-10 dataset is a collection of 60000 32x32 colour images that partitioned into 10 distinctive classes, 50000 for training and 10000 for testing. Each class of image consists exactly 1/10 of both parts. Noting that the classes are completely mutually exclusive and no overlap between one and another.

B. Dataset Preprocessing

For dataset preprocessing, we first added noise to the images for better performance by randomly treating images with crop, flipping, and adjusting hue, contrast and saturation. After that, we splitted the dataset by different size of batch to reduce process time.

IV. Proposed approach

A. Models Description

1) Multilayer Perceptrons: A multilayer perceptron (MLP) is a deep, artificial neural network. It is composed of more than one perceptron. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that handle the data through neuron-like processing which refers to the recognition of the divison of the image. The perceptron, an algorithm that classifies (a classifier) input by separating two categories with a straight line, is the basic unit for MLP. Input is typically a feature vector x multiplied by weights w and added to a bias b: y = w * x + b.A perceptron produces a single output based on several real-valued inputs by forming a linear combination using its input weights. Here is the math formulation to the process:

$$y = \phi(\sum_{i=1}^{n} w_i x_i + b) = \phi(w^T x + b)$$

where w denotes the vector of weights, x is the vector of inputs, b is the bias and ϕ is the non-linear activation function.[5] MLP comprises multiple above computation.

a) Backpropagation: Backpropagation is a widely used algorithm in training MLP.The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule.In MLP the overall network is a combination of function composition and matrix multiplication:

$$g(x) := f^L(W^L f^{L-1}(W^{L-1}...f^1(W^1 x)...))$$

where x:input (vector of features), y:target output, C:loss function, L:the number of layers, W^l:the weights between layer l1 and l,f^l:activation functions at layer l.For a training set there will be a set of inputoutput pairs, (x_i, y_i).For each inputoutput pair in the training set, the loss of the model on that pair is the cost of the difference between the predicted output g(x_i) and the target output y_i: C(y_i,g(x_i)).Then each individual component of the gradient, $\frac{\partial C}{\partial w_{jk}^l}$, can be computed by the chain rule.[6]

b) Mini-batch Gradient Descent: Mini-batch gradient descent is a variation of the stochastic gradient descent algorithm (SGD) that splits the training dataset into small batches that are used to calculate model error and update model coefficients.Instead of going over all examples, Mini-batch Gradient Descent sums up over lower number of examples based on the batch size. Therefore, learning happens on each mini-batch of b examples:$w = w - \alpha \nabla_w J(x^{i:i+b}, y^{i:i+b}; w)$ where i is the number of epochs.Partition the training data set into b mini-batches based on the batch size. If the training set size is not divisible by batch size, the remaining will be its own batch.[7]

2) Convolutional Neural Networks: A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.The specific process is as follow: taking a small matrix of numbers (called kernel or filter) and passing it over the image and transform it based on the values from filter, subsequent feature map values are calculated according to the following formula, where the input image is denoted by f and our kernel by h.

$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$; then padding the convoluted image with an additional border to resist shrinking; fro the next the pooling layer is responsible for reducing the spatial size of the convolved feature and decreasing the computational power required to process the data through dimensionality reduction; finally adding a fully-connected layer for learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. After this general process,the input image has been converted into a suitable form for MLP. [8]

B. Process and Architecture

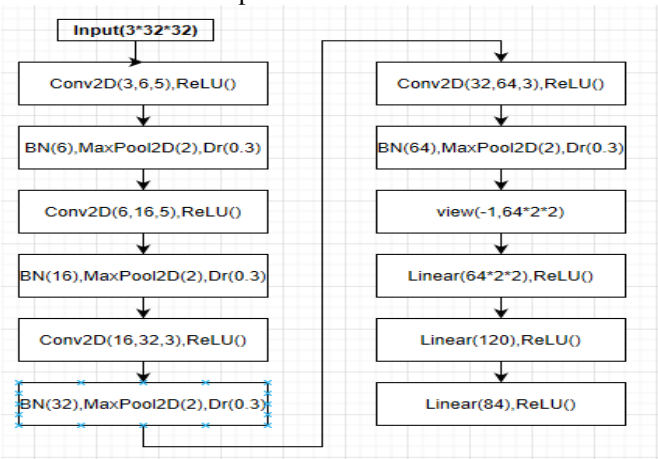
1) Multilayer Perceptrons: In the preparation for MLP, we defined four activation functions (sigmoid, relu, leakyrelu, softmax) for forward and backward layer, mini-batch gradient descent for backpropagation and standardized image data by Normalizer from sklearn.preprocessing for speeding up training and archiving higher accuracy - lower loss.

The process of neural network began with initializing the layer dimension ,weights and bias. The three parameters (number of layer, batch size and activation function) differ in different experiments in our project, so one of the parameters is adjusted before each forward pass. For example, if we change the number of layer from 2 to 3, the units for each layer will be from [[3072,128],[128,64],[64,10]] to [[3072,169],[169,128],[128,64],[64,10]], the dimension for the original layers, batch size and activation functions(i.e from ReLU, ReLU to ReLU, ReLU, ReLU) remain the same. The forward pass refers to calculation process which is consist of the basic computation steps explained in the section of 'Model Description' where the variables are forward propagated through the network. It traverses through all neurons from the first to the last layer.

In the backward pass, three parameters remain the same as the forward pass. The backward pass refers to process of counting changes in weights (de facto learning), using mini-batch gradient descent algorithm in this project .Neural networks are trained during the process: the partial derivative of the loss function are calculating with respect to each parameter inherited from the above by using the inductive nature of the chain rule. Then updating the weights and bias by the gradient, the learning rate involved is fixed to 0.01 and epochs are 1000 during this hyperparameter tuning procedure.

2) Convolutional Neural Networks: The CNN model is learned from a PyTorch tutorial.A package called torchvision which has a data loader for CIFAR10 is used for this model-built section. Since we regarded batch norm, number of convolutional layers and dropout as the parameters should be tuned, we modified the original code by adding two more additional convolutional layers and batch normalization.

The following graph shows the CNN architecture translated from the adjusted code, where 'BN' refers to batch norm and 'Dr' refers to dropout.



This architecture consists of an input layer with four combinations of one 2D Convolutional filter layer with ReLU activation function and one MaxPool2D with a dropout layer. The former two convolutional layers 'conv1' and 'conv2' both apply a 5*5 kernel with stride=(1, 1) that is the kernel for MaxPool has size of 2 with stride=2 and the padding is 1, while the kernels for the later two are 3*3–since additional depth was added, the kernel sizes had to be decreased from 5x5 to 2x2 or 3x3, otherwise we would get a CUDNN STATUS BAD PARAM error describing that computation was attempted on NULL values. The dropout is fixed to 0.3 in this case (there exists 2 values 0, 0.3 for dropout in different case). There is a batch normalization following each 'conv' which improves training efficiency by normalizing data for each training batch at each layer of the network. The mean and variance are independently calculated for each dimension. We used ReLU as an activation function because it helps speed up gradient descent as opposed to a tanh or sigmoid function. It can be also noted that the Flatten() filter before the first fully connected layer is a call to PyTorchs reshape(size, -1) function that flattens an array from 2D to 1D where size is the current size of the image within the network. The Flatten() filter follows by three linear function layer (fully connected layer) with ReLU activation function which are similar to the process of MLP.Cross Entropy (nn.CrossEntropyLoss()) is applied as the loss function and Adam (optim.Adam()) for optimization. These criterion and optimization functions are selected because Cross Entropy Loss is popular for classification problems and Adam is a fast and popular optimizer.

C. Algorithm Selection and Implementation

The model with better performance is selected by hyper-parameter tuning and the architecture adjusted in the field of number of hidden layers, batch size and activation function for MLP, while for CNN the adjusted parameters are number of convolutional layers, number of batch normalization, batch size and dropout. Since the class accuracy achieved by CNN exceeds the performance of MLP with different parameters/architecture among nearly all classes by over 20%, CNN is selected for the following optimization process.

D. Model Optimization

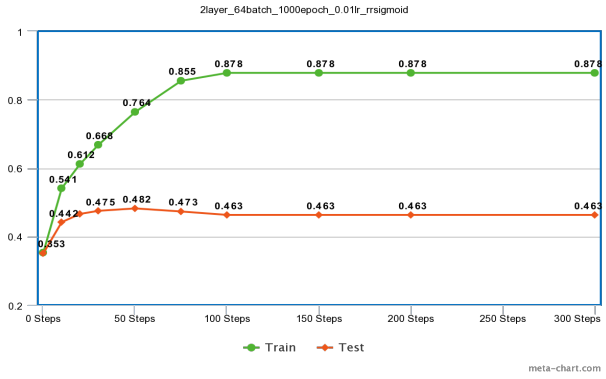
Generally, CNN performs better than MLP in all classes even though MLP is adjusted multiple times in the realm of hyper-parameter and architecture. However, it should be noticed that the run-time for training CNN is extremely high which lasts for hours and loss decreases/ accuracy increases slowly along with the epoch. To explore a quicker convergence, we did related research on the fast convergence for CNN under different cases, which refers to the approximation to some particular level of loss within the epoch/iteration as small as possible. One of papers solves the problem to some extent in a relatively easy method. The paper describes a phenomenon, which is named as "super-convergence", where neural networks can be trained an order of magnitude faster than with standard training methods. One of the key elements of super-convergence is training with one learning rate cycle and a large maximum learning rate. A primary insight that allows super-convergence training is that large learning rates regularize the training, hence requiring a reduction of all other forms of regularization in order to preserve an optimal regularization balance.[9] Due to the above reasons, we will increase the learning rate from 0.01 to 0.1 for CNN and compare this result with the data obtained by different parameter-adjusting.

Another paper suggests that larger batch size attains a lower loss value early in the training for neural networks[10] whose author chooses a batch size of over 500 in the experiment. Nevertheless, our project is under the requirement of the mini batch case, we will test the learning rate 0.1 with batch size 128/256 and place the larger scale improvement in the further exploration.

V. Results

A. Comparison between Training Accuracy and Test Accuracy

The training-test comparison graph for the experimental 2layer-64batch-1000epoch-0.01learning-no dropout- rate-relu/relu/sigmoid by applying MLP is shown below.

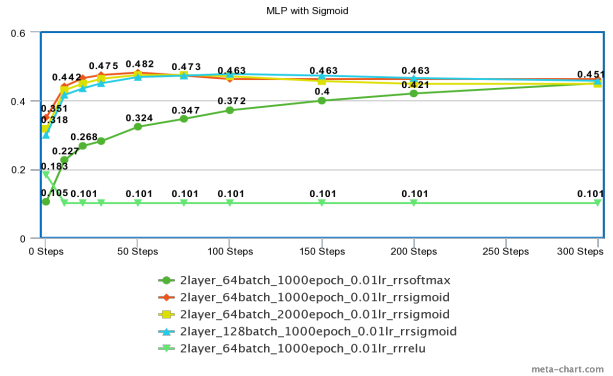


The gap between the training and test accuracy indicates the amount of overfitting. There is a strong overfitting in this group. This may result from small batch size(64) and 0

dropout which shows the necessity of regularization in the contrary.

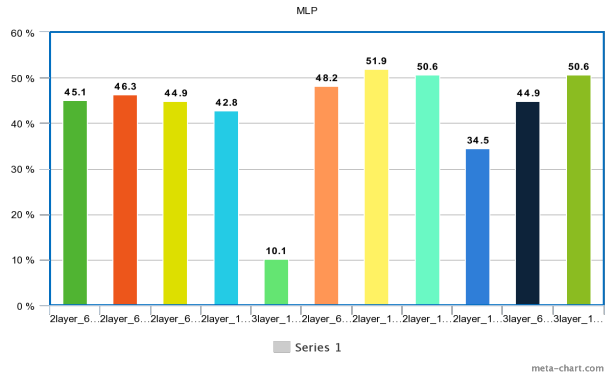
B. Hyper-parameter Tuning and Architecture Adjustment for MLP and CNN

a) MLP: The line graph below mainly shows the effect of activation function. The term 'rrsigmoid' refers to the case that the activation between all four layers (input layer, two hidden layers, output layer) are ReLU, ReLU, Sigmoid.



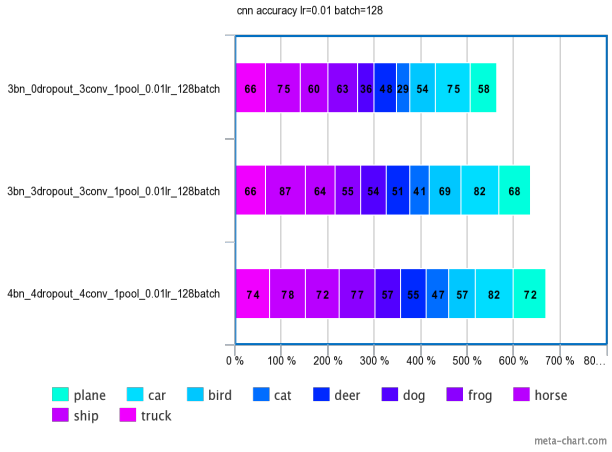
The groups with condition 'rrsigmoid' converge fast with 50 steps (each step refers to one forward pass plus one backward pass), while 'rrsoftmax' converges slower but achieve similar accuracy. The group with all ReLU converges fastest but dives into a poor accuracy result. The limitation for ReLU is that it should only be used within Hidden layers of a Neural Network Model[11]. Hence for output layers we chose to use Softmax/Sigmoid function for a Classification problem to compute the probabilites for the classes.

The following histogram graph demonstrates the accuracy to 11 groups with different parameter setting for MLP. The descriptions to setting are in Appendix.



Performance of all groups by MLP are barely satisfactory. The group with'2layer-128batch-1000epoch-0.01lr-leakyrelu-leakyrelu-softmax' archives the highest accuracy with 51.9% which shows larger number of samples propagated through the network and the softmax before last layer improve the accuracy to some extent.Group 2 and 10/ group 4 and 11 differ in the number of layer– 2 and 4 are with 2 hidden layers while 10 and 11 are with 3 layers, but 2 performs better than 10, 4 is worse than 11. The effect of the number of layer on the accuracy is hard to pin down.

b) CNN: The test result for CNN is shown in the graph below, where each column for the experimental group is consist of the accuracy for each class. The parameter setting for each group is listed on the y-axis of the graph, in which 'bn' refers to the number of batch normalization and 'conv' refers to the number of convolutional layer. Remind that the parameters adjusted in this section are the number of convolutional layer, batch size and dropout, learning rate is fixed to 0.01 and activation function is always ReLU.

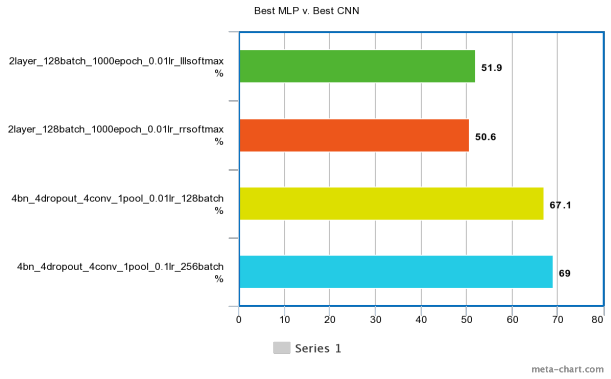


It can be noted that the group with 0 dropout performs worst with an average accuracy of 57%, falls behind the group with the same number of convolutional layer and the same batch size by 7%. This result confirms the fact that overfitting occurs with a high probability if there is no dropout for regularization(we also find that the accuracy during training for this group is above 90 % in the line graph '0dropout' shown in Appendix).

It can be found from the above graph and histogram in Appendix that the setting with 4 convolutional layers performs relatively better compared to the group with only 3 of that and the same batch size of 128. To some extent, this phenomenon shows that more convolutional layer can increase the accuracy, this may be contributed by more features extracted, larger number of weights in the network and the higher model complexity. But there is a limitation. Above it, instead of extracting features, the model tends to overfit the data. Additionally, the group with the batch size of 128 even performs better than the group with the same number of convolutional layer but 256 batch size, the similar result is found in the quote of Kaiming He from Microsoft Research about reducing batch size: "To fit the 1k-layer models into memory without modifying much code, we simply reduced the mini-batch size to 64, noting that results in the paper were obtained with a mini-batch size of 128. Less expectedly, the results with the mini-batch size of 64 are slightly." This abnormal deserves further research before drawing an explanation. What is interesting to be noticed is that nearly all groups receive higher accuracy in the class of artificial objects(ship, car, plane) than animals(cat, dog, deer) which may due to the regularity of the former and the similarity between the latter(cat and dog) or complex

appearance(antler) for learning to recognition. Some groups even archive over 0.9 accuracy when recognizing 'Car'. More specific graphs (or with other parameter settings) are in Appendix.

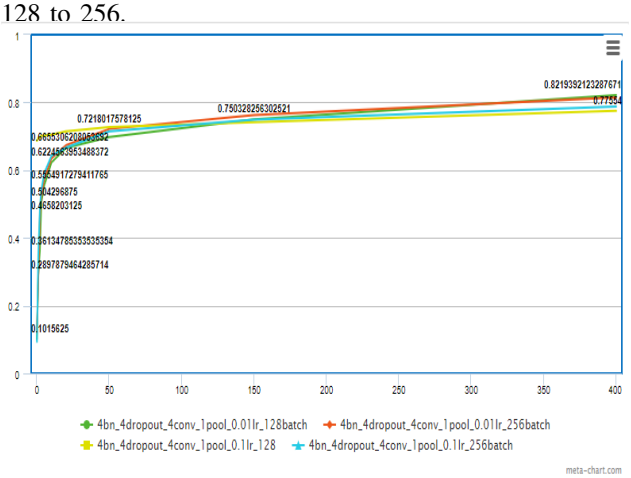
C. Performance of Different Model



We picked two best-performance groups from MLP and CNN respectively where the accuracy is calculated by the average performance for 10 classes. Obviously, CNN exceeds MLP by approximately 20%.This advantage is caused by layers of convolution and pooling to some extent.Convolutional layers take advantage of the local spatial coherence of the input. For images, this can be seen by the fact that the image loses its meaning when the pixels are shuffled. Using this property, CNNs are able to cut down on the number of parameter by sharing weights. This makes them extremely efficient in image processing, compared to multi-layer perceptrons.

D. Convergence Fast fro CNN

CNN performs better than MLP but the training time for the former is extremely long. Based on the theory One-Cycle Policy from the section 'Model Optimization', we increased learning rate from 0.01 to 0.1, batch size from 128 to 256.



Since the scale of epoch is large(the basic unit is 1000), the line with 0.1 learning rate boosts up (or say converges to some level) in a flash which is in line with our expectations– with a relatively small learning rate, the network begins to converge and, as the learning rate increases, it eventually becomes too large and causes the test accuracy/loss to converge suddenly[10]. While the effect of larger batch size (blue line and red line) does not demonstrate an obvious difference with the line set as relatively low learning rate/ batch size(but they still grow slightly faster than the green one, the fact meets the theory "a larger batch size results in a fast convergence" as well). The optimal increasing extent deserves a further research.

VI. Discussion and Conclusion

Generally the performance of two models and the adjustment to the speed of convergence for CNN fit in the expectation. However, the phenomenon of overfitting occurred when we tested some groups of MLP and CNN. The solutions like data augmentation and early stopping should be considered for the further research.

The further exploration will concentrate on the optimization of "super-convergence"/convergence-fast method where neural networks can be trained an order of magnitude faster than with standard training methods. We boosted the learning rate of CNN model from 0.01 to 0.1 in the experiment, but the optimal increasing extent deserves a further research. The paper cited derives a simplification of the Hessian Free optimization method to compute an estimate of the optimal learning rate which can be considered as a solution to this[9]. In addition, the effect of momentum and learning rate on the training dynamics is valuable to be noticed as well since it is designed to accelerate network training and momentum and learning rate are dependent on each other.

References

[1] Image classification. Retrieved from https://www.tensorflow.org/lite/models/imageclassification/overview

[2] Brownlee.J (2018), When to Use MLP, CNN, and RNN Neural Networks. Retrieved from https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

[3] Arendt.K (2019), CIFAR-10 images: classification using MLP vs. CNN. https://krzysztofarendt.github.io/2019/01/29/cifar-10.html

[4] Abdelfattah.A (2017), Image Classification using Deep Neural Networks A beginner friendly approach using TensorFlow. https://medium.com/@tifa2up/image-classification-using-deep-neural-networks-a-beginner-friendly-approach-using-tensorflow-94b0a090ccd4

[5] ROSENBLATT.F (1958), THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN, Vol. 65, No. 6, 19S8.

[6] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "Back-Propagation and Other Differentiation Algorithms". Deep Learning. MIT Press. pp. 200220.

[7] Dabbura.I (2017), Gradient Descent Algorithm and Its Variants

[8] Saha.S (2018), A Comprehensive Guide to Convolutional Neural Networks the ELI5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

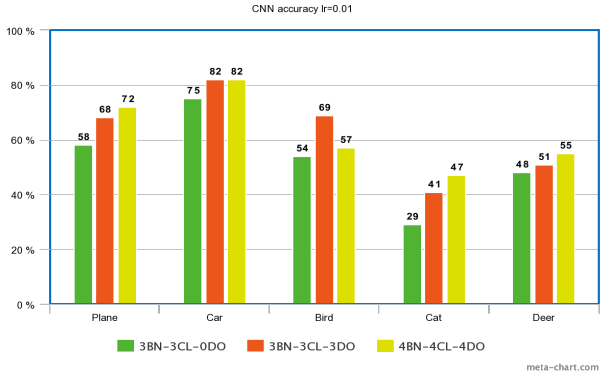
[9] Leslie N. Smith, Nicholay Topin (2017), Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. Cite as: arXiv:1708.07120 [cs.LG]

[10] Gupta.A (2018), One-cycle policy. Retrieved from https://towardsdatascience.com/https-medium-com-super-convergence-very-fast-training-of-neural-networks-using-large-learning-rates-decb689b9eb0

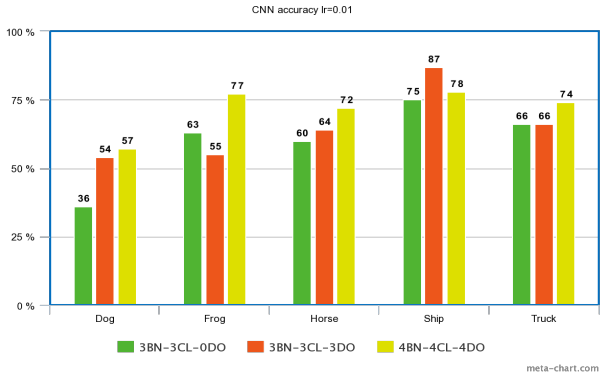
[11] Walia.A.S (2017), Activation functions and its types-Which is better? https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

VII. Appendix

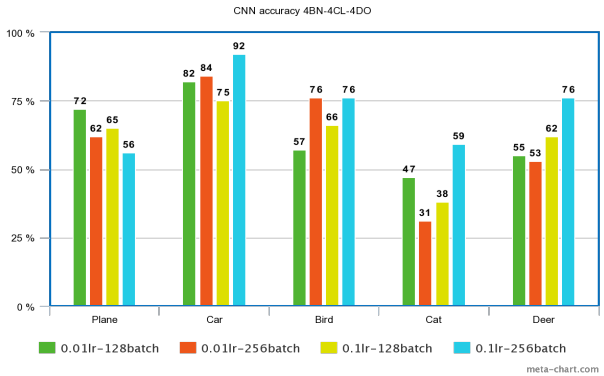
Histogram for the accuracy of 3/4 convolution-layer CNN on the former 5 classes:



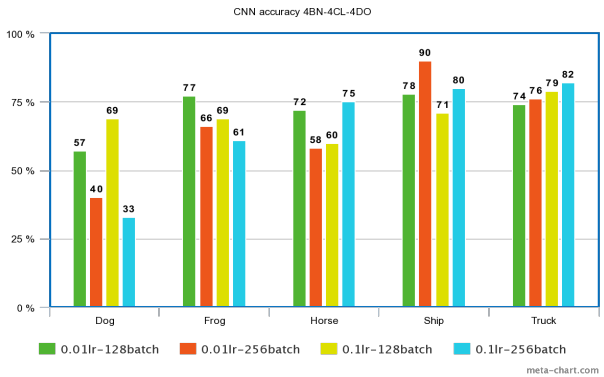
Histogram for the accuracy of 3/4 convolution-layer CNN on the latter 5 classes:



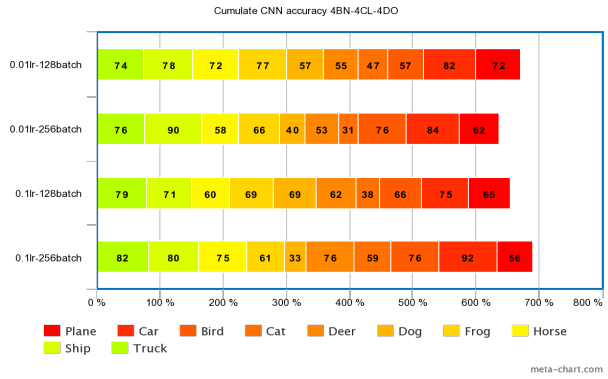
Histogram for the accuracy of 4 convolution-layer CNN on the former 5 classes:



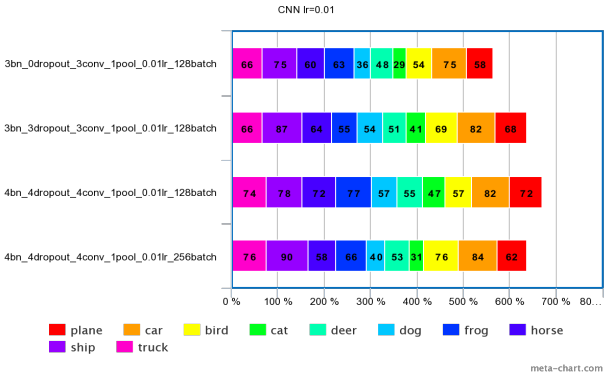
Histogram for the accuracy of 4 convolution-layer CNN on the latter 5 classes:



Cumulate CNN accuracy 4BN-4CL-4DO:



Cumulate CNN accuracy when lr=0.01:



Annotations to the histogram of different groups by applying MLP:

```
left 2layer_64batch_300epoch_0.01lr_rrsoftmax
     2layer_64batch_1000epoch_0.01lr_rrsigmoid
     2layer_64batch_2000epoch_0.01lr_rrsigmoid
     2layer_128batch_1000epoch_0.01lr_rrsigmoid
     3layer_128batch_1000epoch_0.01lr_rrrrelu
     2layer_64batch_1000epoch_0.01lr_rrsoftmax
     2layer_128batch_1000epoch_0.01lr_lllsoftmax
     2layer_128batch_1000epoch_0.01lr_rrsoftmax
     2layer_128batch_1000epoch_0.01lr_sssoftmax
     3layer_64batch_1000epoch_0.01lr_rrrsoftmax
right3layer_128batch_1000epoch_0.01lr_rrrsoftmax
```

One CNN group without dropout compared to the group with 0.3 dropout :