Multi-class Image Classification using Deep Neural Networks on Extremely Large Dataset

Marija Stanojevic & Ashis Kumar Chanda
CIS Department, Temple University
{marija.stanojevic, ashis}@temple.edu

Multi-class image classification problem is a popular research topic nowadays. Many researchers have proposed different deep neural network models to solve this problem in different ways. However, the problem is still challenging for classifying thousands of classes. In this paper, a multi-class image classification problem is solved for a extremely large dataset. The large dataset contains 5,270 product images for a french e-commerce site containing almost 6.4 TB of image data. To solve this multi-class classification problem, different existing models, such as vanilla convolutional network, dense network, residual network, aggregated residual transformation and wide residual network are modified and tested. Adjusted wide residual network provides the best accuracy result (41.64%) over randomly generated validation dataset. Our code is publicly available on online for future research purposes [1].

1. Introduction

Image classification is the task of assigning an input image to a label from a set of categories. Even though the problem looks simple and humans have a very good performance on this task, there are many challenges in learning to a machine to classify images [3]:

1. Viewpoint variation (object can take different positions in respect to camera)
2. Scale variation (object size variation depends on its closeness to camera and camera settings)
3. Deformation (deformation or shape change)
4. Occlusion (parts of an object can be occluded)
5. Illumination conditions (different lightening of pixels)
6. Background clutter (object blend into environment)
7. Intra-class variation (objects in class differ drastically)

Many other problems (image segmentation, object detection) in computer vision can be reduced to image classification problems, giving even more importance to this problem. Image classification can be decomposed into three tasks: 1) features extraction and selection; 2) features/images embedding; 3) image classification (supervised, semi-supervised, unsupervised).

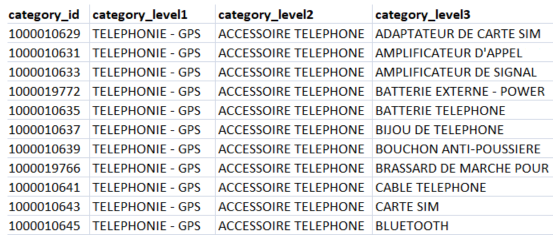
Usually, first, two steps have been done with scale-invariant feature transform (SIFT) or in other hand-crafted ways to produce vector descriptors of significant image features. In the last step, k-nearest neighbor (KNN), support vector machine (SVM), or other classification method was applied on image feature vectors to categorize images. The final step can also be done in an unsupervised way.

The application of deep learning methods has rapidly increased in the current decade and is growing with time so that now machine can make a decision correctly by analyzing image, voice, or steaming data. The applications in computer vision spread on many research tasks, such as object detection, object and image classification, face recognition, and so on.

The dominant model for image classification is convolutional neural networks (CNN) because it can learn from end-to-end training and achieve good accuracy. A CNN can do all tasks using gradient descent (GD) or other optimization methods to optimize the steps for the best final result. However, their usage wasn't popular in the past because of large amount of training data, and computing resources were required. In the last decade, both problems are solved, and different modifications of CNN are created to solve image classification problems.

To improve the accuracy of the solution, researchers proposed to gradually increase the number of layers in CNN. For example, LeNet5 [17] proposed to use five layers, ImageNet [11] has eight layers, VGG model [13] came with 19 layers, and finally, 22 layers are used in Inception/GoogleNet [5]. Recently, changes are introduced that allowed more than 100 layers to be trained in a reasonable time by using Highway [15] and Residual networks (ResNet) [7].

[1] https://github.com/marija-stanojevic/image-classification



category_id	category_level1	category_level2	category_level3
1000010629	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	ADAPTATEUR DE CARTE SIM
1000010631	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	AMPLIFICATEUR D'APPEL
1000010633	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	AMPLIFICATEUR DE SIGNAL
1000019772	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BATTERIE EXTERNE - POWER
1000010635	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BATTERIE TELEPHONE
1000010637	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BIJOU DE TELEPHONE
1000010639	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BOUCHON ANTI-POUSSIERE
1000019766	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BRASSARD DE MARCHE POUR
1000010641	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	CABLE TELEPHONE
1000010643	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	CARTE SIM
1000010645	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BLUETOOTH

Figure 1. A sample list of product classes

Data sets	Size	# product images
Train data	58.2 GB	7,069,896
Test data	14.5 GB	1,768,182

Table 1. Summary of data sets

Categories	Label number
Category 1	49
Category 2	483
Category 3	5270

Table 2. Number of labels in each categories

As the number of layers increased, the problem of exploding/vanishing gradient became substantial, and training became less efficient. This challenge was largely addressed using normalization layers and normalizing input [9]. However, another problem appeared: degradation (as the number of levels increases, accuracy gets saturated and then starts degrading), which is not caused by over-fitting.

Multiple techniques were created to overcome this problem based on two main ideas: 1) bypass signal from one layer to some future layer via identity connection and 2) widen layers instead of increasing depth for better results. This project uses ResNet [7], Wide Residual Networks (WRN) [18], ResNext [16] and DenseNet [8] as starting points that are modified to find the best solution for problem presented in section 2.

The rest of the report is organized as following - data set is presented in section 2.1, and the problem is described in section 2.2. Related works are discussed in section 3, and proposed methods are described in section 4. Experimental results are shown in section 5, and a discussion on results is given in section 6 and finally. Summary and conclusions are in section 7.

2. Dataset

For this study, we take data from a Kaggle competition [2]. The competition is designed by C'discount company [1] that owns a popular non-food e-commerce site C'discount.com in France. The company is a general-purpose retailer that has many different product categories. Therefore, this website asks for a machine learning solution that can automatically classify the products based on their images. To train the model, they provide a data set of 7,069,896 million products that can contain one to four images per product, where each image has a size of 180 x 180 pixels in RGB format.

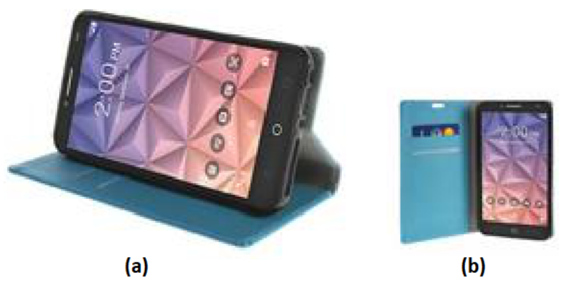


Figure 2. Deformation and scaling problem in product images

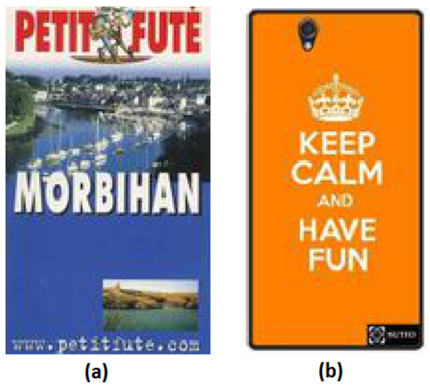


Figure 3. Occlusion problem in product images

In this dataset, each product has three levels of categories. The first contains 49 labels, the second has 483 labels, and the third has 5,270 labels, shown in Table 2. Category levels are organized in a tree-like structure, and the task is to classify objects into one of 5,270 labels of the third level. Figure 1, shows a sample list of product classes where the first level category is TELEPHONE-GPS and the second level category is ACCESSOIRE TELEPHONE containing many third-level categories. Since these items belong to the same root category, they should have the same shape and attributes. However, as it is shown in Figure 1, there are very different objects under the same root category. Hence, it brings a challenging problem to apply the machine learning method and develop the best architecture. The details of the training and test data set are shown in Table 1.

The Kaggle competition provides data set in BSON format (binary string format). Since it is important to look at the data to understand the problem challenges, a program [4] is used to convert BSON files into image format. Figures 2, 3 and 4 are some examples of the dataset that clearly show many of the image classification challenges.

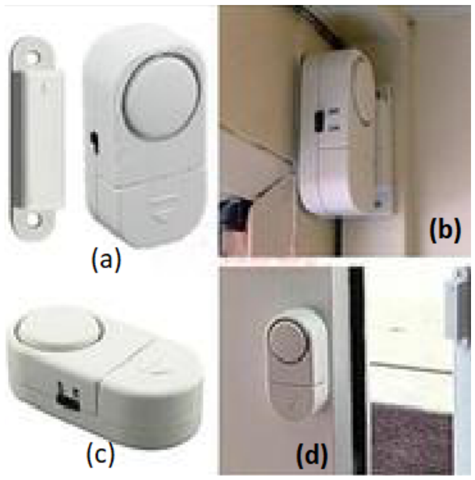


Figure 4. One item in different angles and backgrounds

3. Problem description

This project aims to classify products into one of the 5,270 categories based on the images posted by the company. As explained in the above section, the real-life dataset is very interesting because of many classes and a huge amount and variety of images.

Overview of commonly used data sets for image classification problems is given in Table 3 and all these are much smaller in the number of items and number of categories than our selected data set. Recently, new data sets are in development, such as ImageNet and COCO, which are more similar to our selected dataset, but they are developed to serve multiple computer vision problems.

Besides the size and number of classes, our data is difficult since it contains all challenges presented in Introduction. Some products can have multiple images from different viewpoints, while other products have only one image, and it can be taken from any viewpoint (not standardized), as shown in Figure 4. Objects on images vary in scale (Figure 2 (b)) and sometimes are shown in different sizes on the same image or product is shown in its parts. Object can have different deformations (opened and closed laptop, packed and unpacked clothes,...) (Figure 2).

In some cases, products are occluded: screen of the phone which shows working display (Figure 2) or text or declaration note above the object. The shapes and text background of Figure 3(a) and (b), look similar, but these are two different products (book and mobile case). Hence, it becomes challenging to make a decision correctly. Even though images are expected to be as clear as possible for the purpose of sale, they are taken from different users over time, and therefore distinct illumination conditions have to be taken into consideration as well.

Some images contain products on white background, but it's often not the case. There are occurrences where background has similar color/pattern as object. Such example is given on Figure 4 (b) and (d). Since products are general-type objects, like chairs, clothes, and so on, they have extensive intra-class variation.

data sets	Classes	# images (train + test)	image size
CIFAR-10	10	50000 + 10000	32x32
CIFAR-100	100	50000 + 10000	32x32
SVHN	10	73257 + 26032	32x32

Table 3. Summary of data sets

4. Related Works

Image classification problem is an active area of research, and most commonly, it is done using different convolutional networks. In this section, basic CNN model is described at first since that concept was used as a baseline. After that, other recently developed methods used as a starting point for solving this problem, and they are described next.

4.1. Convolutional Neural Network (CNN)

The idea of CNN was first discussed in 1980 [6], but it became popular much later. CNN architecture is very similar to common neural network, but it allows better encoding of certain properties of image file. To retrieve the properties, CNN connect each neuron to only a local region of the input image volume; rather than whole image. This local space is known as respective field and the process of reading whole image file through this local space scanning into next layer of neurons is known as filtering. Since CNN use the same weight vector for all neurons in a single depth slice, the forward pass of the CNN layer can in each depth slice be computed as a convolution of the neurons weights with the input volume. Hence the name is convolutional layer.

In CNN architecture, we can find three types of layers: convolutional layer, pooling layer, fully connected layer that can be additionally accompanied with batch normalization or dropout. These layers help to transform image volume into output (class score). However, the accuracy of final outcome depends on the number of layers, their combination, size of layers, ways layers are connected, normalizations and activation functions used. Hence, it is our objective to use CNN in our problem with different structures and find a model that would result in good accuracy. We will use different modifications of basic CNN as the state-of-the-art methods, but our baseline will still be a plain CNN described in this section.

4.2. Residual Network (resNet)

Residual networks [7] are created as a solution to degradation of very deep CNN, concretely with focus on improv-

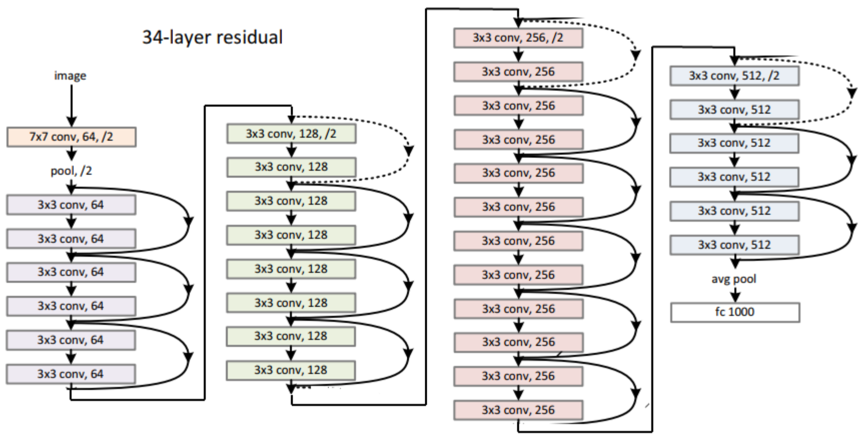


Figure 5. Simplest resNet example given in paper - 34 layers

cardinality	1	2	4	8	32
width of bottleneck d	64	40	24	14	4
width of group convolution	64	80	96	112	128

Table 4. Timeline of activity

ing VGG-34 network. This network is connected in such a way that it doesn't learn to optimize original mapping, but it rather optimizes residuals (Figure 5). It consists of many blocks of residual learning connected in a deep CNN.

Each block consist of two 3x3 convolution layers with ReLu activation between them and identity link that is added to the output of the block. Therefore block learns residuals $F(x) = H(x) - x$ where $F(x) = W_2 * \sigma(W_1 x)$ and W_1 and W_2 are weights of first and second layer in a block, respectively. Finally, $y = F(x, W_i) + W_s * x$, where W_s is a projection of input and in case of identity it doesn't exist. Even though results from resNet paper suggest that there is a slight improvement in results when projection W_s is used, authors suggest avoiding it in order to decrease number of parameters drastically.

Residual networks are first networks that have been trained with more than 1000 layers, but in case of such high number of layers authors suggest modification of residual block that will decrease number of parameters. They use three layers in block in that case, where first and third layers have 1x1 convolutions and represent "bottlenecks" and middle layer is 3x3 convolution of features that are created in "bottleneck".

Results are evaluated on ImageNet and CIFAR-10 data sets and show improvement in performance comparing to non-residual networks even when number of parameters is comparable (depth is still larger in resNets in that case).

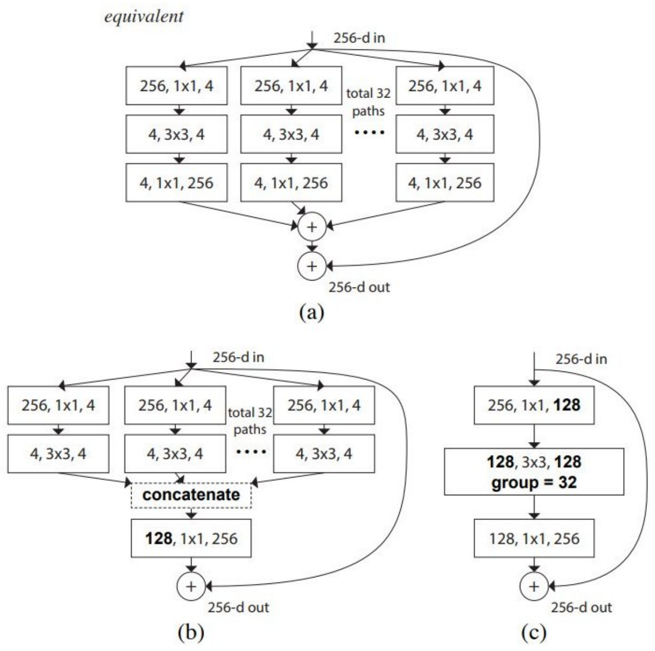


Figure 6. ResNext block in multiple views

4.3. Aggregated Residual Transformation - resNext

Idea of ResNext [16] network is based on residual networks, except that instead of widening the layers, they separate them in C sub-layers which are trained in parallel and summed up before summation with the identity link. Number of parallel blocks is called cardinality (C) and it's relation to layer width is given in Table 4. Block can be presented in multiple different ways as shown in Figure 6. Reformulations produce nontrivial topologies only when the block has depth bigger than 2.

Results of resNet and resNext networks per epoch are given in Figure 7 and they show that error drops like stairs and at least 30 epochs are required for a first bigger drop in

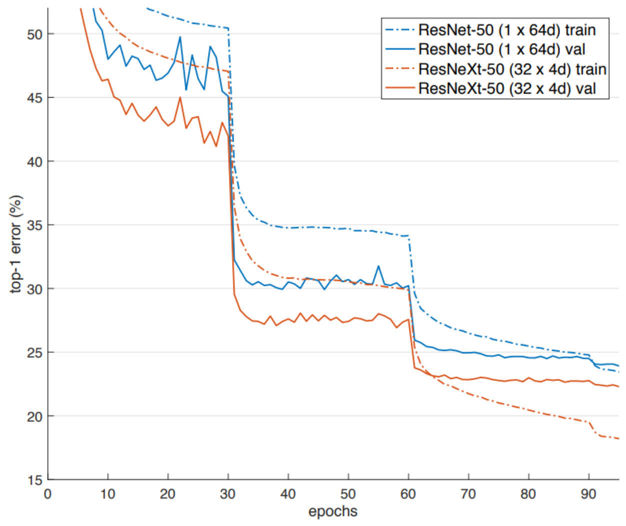
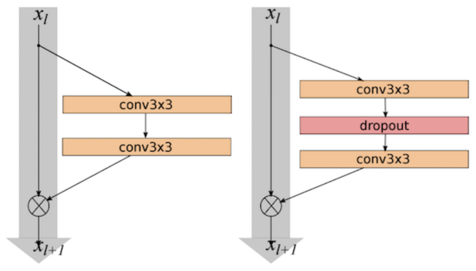


Figure 7. Training and test error for resNet and resNext per epoch number



(c) basic-wide (d) wide-dropout

Figure 8. Simplest resNet example given in paper - 34 layers

error.

In resNext $F(x) = \sum_{i=1}^{C} T_i(x)$, where $T_i(x)$ can be an arbitrary function which projects x into an embedding and then transforms it. Then, $y = x + \sum_{i=1}^{C} T_i(x)$. Authors don't mention or consider dropouts in their work and they don't discuss projections of identity links as they are given in resNets. Results show that resNext has better performance than resNet for the same number of layers and features. When comparing two resNext networks, better results are achieved with higher cardinality C keeping number of parameters the same. Also, results show that for similar number of parameters resNext has better results than WRN on CIFAR-10 and CIFAR-100 data sets.

4.4. Wide Residual Network (WRN)

Wide residual networks [18] are improvement of resNet networks that uses similar residual blocks with idea of widening them instead of concatenating a lot of them to get a deep network. They test different versions of residual blocks and conclude that best results are produced with blocks that have 2 layers with 3x3 convolutions (Figure 8).

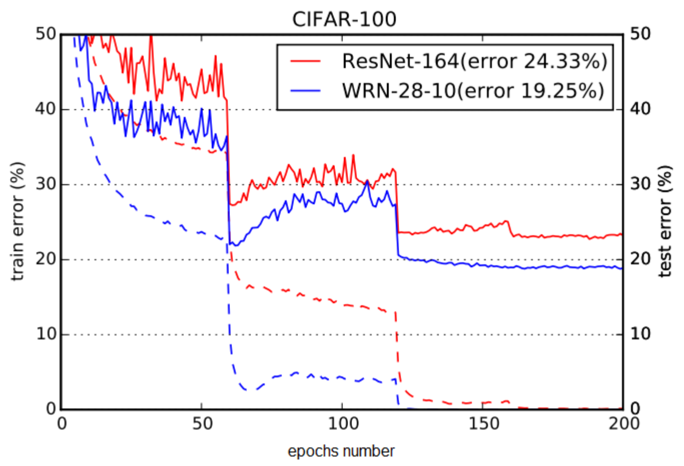


Figure 9. Train and test error for wide residual network per epoch number

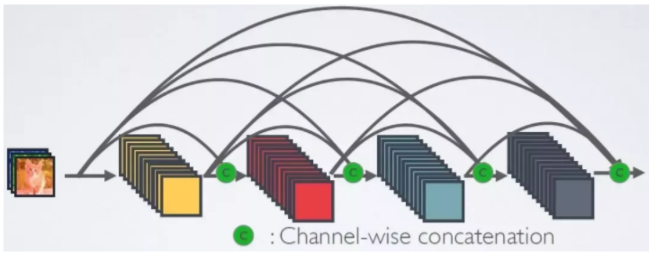


Figure 10. Small denseNet example

The most often widening factor of 2 is used, however results showed that improvement in results is visible when widening factor grows up to 12. Authors didn't test for higher values because of high number of parameters.

Comparing to resNet that doesn't use dropout at all, WRNs suggest that dropout [14] between the 2 layers in residual block gives the best result. They show that this architecture gives better results than resNet with much less layers and similar amount of parameters. Both, resNet and WRN have issues when number of layers becomes too big. In resNet work, they suggest that reason is over-fitting, but they don't give clear results to support that, so research question is if degradation still appears in those cases.

Results of wide residual network per epoch are given in Figure 9 and they show that error drops like stairs and at least 60 epochs are required for a first bigger drop in error.

4.5. Dense Network - denseNet

The authors of [8] proposed denseNet, a new convolutional network architecture, where one layer is connected with all other layers in that block, as shown in Figure 10. Also, input and outputs of each blocks are connected with outputs of all other blocks. This model solves existing problems such as vanishing gradient, feature propagation and reduction of the number of parameters. They also allow

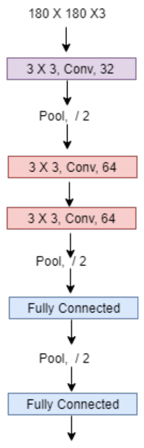


Figure 11. Proposed simple CNN

feature reuse capability throughout the network. The authors showed that their method has improvements in accuracy over previous methods on task of image classification.

Network has L layers which can be described with a non-linear composite function $H_l(X_{l-1})$. Here l indexes the current layer. If output of layer l is added to identity from previous layer which can be described with next formula:

$$X_l = H_l(X_{l-1}) + X_{l-1} \tag{1}$$

However, denseNet uses direct connections from any layer to all subsequent layers. Hence, the l th layer receives the feature values of all preceding layers $(X_0, X_1, ..., X_{l-1})$ as input:

$$X_l = H_l(|X_0, X_1, ..., X_{l-1}|) + (X_0, X_1, ..., X_{l-1}) \tag{2}$$

Here, $|X_0, X_1, X_2, ..., X_{l-1}|$ means the concatenation of the feature values produced in layers $0, 1, 2, ..., l-1$. Hence, there are in total $l \times (l-1)/2$ connections for a l layer network.

For experimental analysis, the authors have conducted experiment on four common object classification data sets (CIFAR-10, CIFAR-100, SVHN, ImageNet). Different values of parameters were used to test the performance of proposed model on these four data sets. The authors have shared the project code (written in Lua) publicly in GitHub.

4.5.1 Memory efficient approach - denseNet

One drawback of denseNet [8] is that it needs huge memory to train the model and propagate the result of one layer to all other layers. The authors of [12] provided a memory efficient solution for this problem which is based on map reducing method. By strategically using shared memory allocations, they reduced the memory cost for storing feature maps from quadratic to linear.

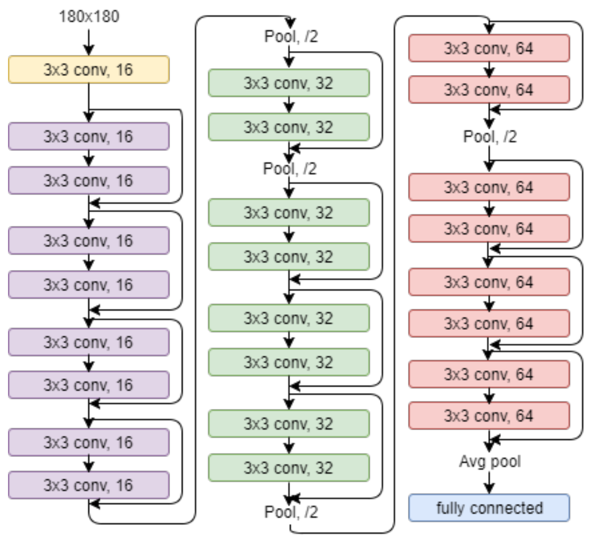


Figure 12. Proposed residual network

5. Methods

In the following five subsections, we describe some models in detail that are used for solving this problem. At the time of designing our proposed models, it was important to keep the same number of parameters between different networks in order to be able to compare their performance. Only the baseline network is smaller in number of the parameters. Additionally, because of the huge data set, the design of the network had to be minimalistic in order to be able to train in a reasonable time and with our resources.

5.1. Convolutional Neural Network

The CNN architecture is used as a baseline in many popular image classification problems [10], [8]. The architecture of CNN model is shown in Figure 11, and it includes three convolutional (Conv), two fully connected (FC) layers, and three max-pooling layers.

Convolutional layers use 3×3 convolutions with filter sizes 32, 64, and 64, respectively. The purpose of pooling layers is to lower the number of parameters since the image size is 180×180 pixels. First, the fully connected layer has a size of 512 units, all activations are ReLu, except in the final layer where the activation is softmax. Adam optimizer is used with categorical cross-entropy loss. The learning rate was 0.001.

5.2. Residual Network

The proposed residual network (Figure 12 is based on the 34-layers residual network as given in the original paper. Because of resource restrictions, this network had to be smaller, so it contains three chunks (different colors in Figure 12 divided by pool layer instead of four blocks that the original resNet has. Additionally, in the second and third

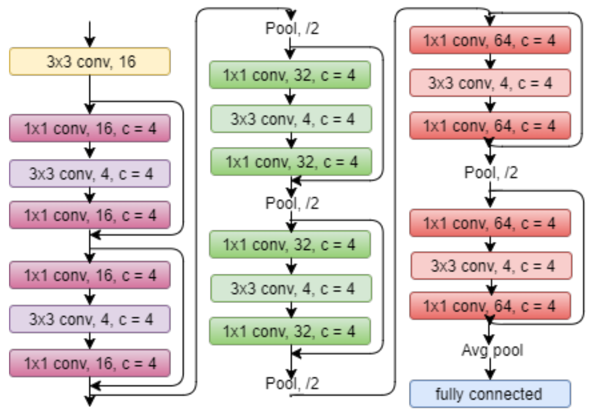


Figure 13. Proposed aggregated residual transformation - resNext

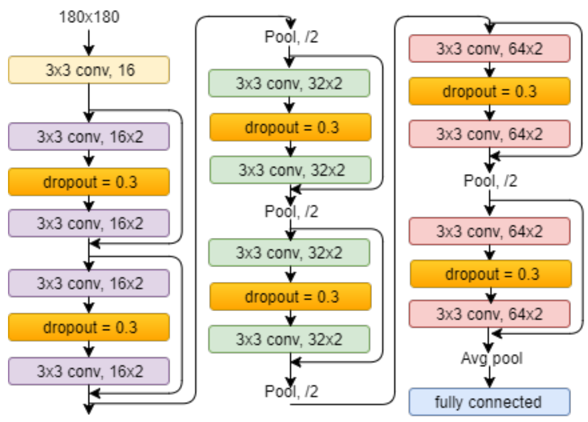


Figure 14. Proposed wide residual network

chunk, there is one more pooling layer after the first residual block. This is not used in original resNet, but pooling helped in reducing the number of parameters and training time. Each chunk contains four residual blocks. Each residual block has two 3x3 convolutional layers.

Since the network is small already, residual blocks with 1x1 bottleneck convolutions surrounding the 3x3 convolutional layer are not used in the proposed residual network. Before all residual blocks, there is a 3x3 convolutional layer, and after all residual blocks, there is an average pool and fully connected layer as used in original resNet. Convolutional layers are smaller (thinner) than in original resNet which helped in reducing number of parameters and training time. Each chunk is twice wider than previous (widths are 16, 32 and 64).

There is no dropout, but after each convolution, there is batch normalization followed by an activation function. Identity is added to output from the last convolution from that block, and then batch normalization and activation function are applied. Pooling layers are implemented as strides of size 2, and they are used exactly before convolution.

In all the layers, ReLu activation function is used which is proven to be the best activation function for this kind of example. Weights are initialized using variance scaling, which was tested together with other weights initialization on similar problem by one of the authors and gave best results comparing to the other weights initialization methods. Biases are initialized with zeros. All convolutions have L2 regularizer. Fully connected layer has softmax activation, the optimizer is momentum (learning decay is 0.1 and decay step is 32000) with categorical cross-entropy loss.

5.3. Aggregated Residual Transformation

The proposed aggregated residual transformation (Figure 13 is based on the original resNext, which is created in such a manner to have a similar number of parameters as proposed in resNet. The proposed network is smaller than the original examples. It has the same general structure as the proposed resNet, except that each chunk has two resNext blocks.

ResNext blocks consist of two 1x1 convolutions and 3x3 convolutions between them. Cardinality is C=32 because it gave the best results in the original network. There is no dropout, but after the first bottleneck convolution and middle convolution, there is a batch normalization followed by an activation function. The output of the last convolution is joined from all 32 parts and added with identity. Pooling layers are implemented as strides of size two, and they are used exactly before convolution. Weight decay is 0.0001 in all cases. Other parameters are the same as described in the fourth paragraph of 5.2.

5.4. Wide Residual Network

The proposed wide residual network (Figure 14 is based on the original wide residual network so that it has a similar number of parameters as the proposed resNet. The proposed network is smaller than the wide residual networks given in the original paper. It has the same general structure as resNet, except that each chunk has two wide residual blocks.

Wide residual blocks consist of two 3x3 convolutions and a dropout of 0.3 between them. Dropout value, convolutions size, and the number of convolutional layers in the block are the same as the original paper because they produced the best accuracy results in the original paper.

After each convolution, there is batch normalization followed by an activation function. Identity is added to output from the last convolution from that block, and then batch normalization and activation function are applied. Pooling layers are implemented as strides of size 2, and they are used exactly before convolution. Dropout layer is added after activation of the first convolution in wide residual block.

The width is k=2, even though the original paper showed

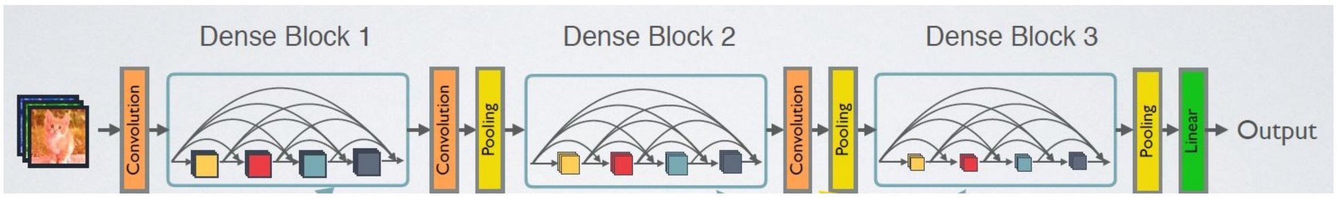


Figure 15. Proposed Dense Network

that k=10 and k=12 give the best results. Using k=10 and k=12 wasn't possible because of the huge number of parameters, so k is chosen from paper to be small and give the biggest improvement in accuracy. Other parameters are the same as described in the fourth paragraph of 5.2.

5.5. Dense Network

The proposed model is based on denseNet [8]. Since each layer of the dense network receives input from preceding layers, the network becomes very large and complex to load in memory. A previous work discussed [12] about how to reduce memory using shared memory strategy, but it wasn't possible in this scenario because the memory access privilege is not available in our system. Hence, thin layers are considered in each dense block for our study.

As discussed first, since the concatenation of huge layers rises a problem in memory, three dense blocks are considered in our proposed method, and only one layer of one block is connected with preceding layers. It means there is no connection between one block layer to another block layer, as shown in Figure 15. In the original architecture, the authors of [8] have used 40, 100, and 160 layers in different dense blocks.

In our proposed method, only $l = 10$ layers are used per block. Each layer consists of three convolutional layers (two bottlenecks with convolutions 1×1 that are surrounding convolutional layer with 3×3 convolution) and an average pool layer with stride 2. It means, the last layer of each block has $10 \times (10-1)/2 = 45$ connections. Since each layer has the same filter size, the transition layer is not considered in the proposed method. But, dropout (0.5) is used at the end of each block.

Before all the dense blocks, there is a convolutional layer with 3×3 convolutions and filter size 16, while all convolutions in the dense block are of size $k = 12$. All the layers use ReLu activation and L2 regularizer. Weights are initialized with the variance scaling method and biases set with zeros. Weight decay is 0.0001. Adam optimizer is used with categorical cross-entropy loss, and the learning rate is 0.001.

6. Experiments and Results

All the state-of-the-art models used in this project are originally implemented in Lua programming language using Torch library. However, we implemented all of the proposed methods in python programming language using tensorflow and tflearn libraries. For resNet, resNext and denseNet, we found some implementations in tflearn, while CNN and wide residual network (including wide residual block) were fully developed by us. Implementations for resNet, resNext and denseNet were checked for accuracy and changed to fit the proposed models.

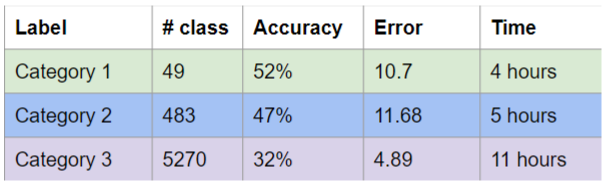
The dataset contains three categories: level 1 category contains subcategories of level 2, while level 2 contains subcategories of level 3. Training (58.19 GB) and test (14.53 GB) data sets contained BSON representations of images. We preprocessed data that consisted of getting products from given data sets and getting images for each product (between 1 and 4 images per product). Additionally, labels were extracted from the training data set, and product ids were extracted from the test data set.

Images were decompressed, and the whole data set was split into a batch of 10,000 products (around 20000 images). We tried multiple sizes of batches, but the decreasing size of batches (128 or 256) increased the time to read and write them twice. On the other hand, batches of size bigger than 10,000 caused memory error. Therefore, 707 training and 177 test batches were created, having 6.4 TB in size after all the changes.

Images were preprocessed and augmented using appropriate functions from the tflearn library. Preprocessing included zero centering and normalization, while augmentation included left-right flip and in case of CNN rotation with max angle 25 degrees.

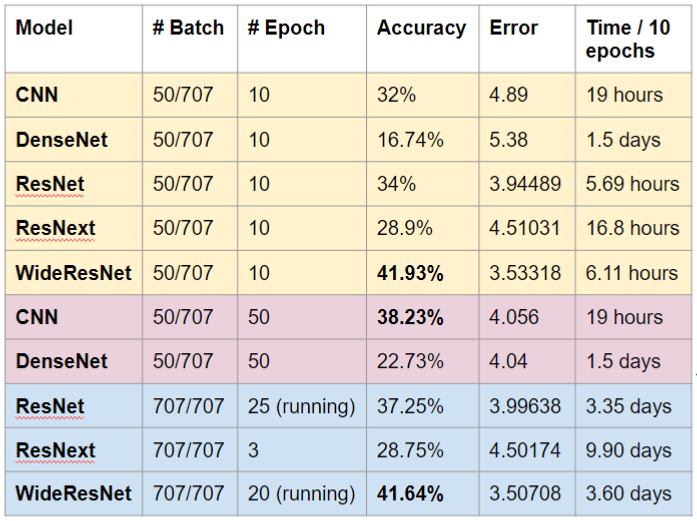
Two cross-validation sets were extracted, one of size 1% of the whole data set, which had around 70000 (1000000) products and another in which 1 product was extracted from each batch, which had around 707 products (1500 images). For both data sets, products were extracted randomly. Unfortunately, the first data set caused memory error, so second cross-validation data set was used.

In order to be able to store a huge amount of data and to process it fast, GPU with big memory and huge storage system was required. For that purpose owlnesttwo high-performance computing (HPC) system was used for dataset splitting and models training and testing. This system has 500 TB of shared storage, NVIDIA Tesla P100 PCIe GPUs with 12 GB of memory, and runs on a Linux operating sys-



Label	# class	Accuracy	Error	Time
Category 1	49	52%	10.7	4 hours
Category 2	483	47%	11.68	5 hours
Category 3	5270	32%	4.89	11 hours

Figure 16. Results of CNN for different category levels on randomly sampled 10 batches that are evaluated on subsample of those 10 batches



Model	# Batch	# Epoch	Accuracy	Error	Time / 10 epochs
CNN	50/707	10	32%	4.89	19 hours
DenseNet	50/707	10	16.74%	5.38	1.5 days
ResNet	50/707	10	34%	3.94489	5.69 hours
ResNext	50/707	10	28.9%	4.51031	16.8 hours
WideResNet	50/707	10	**41.93%**	3.53318	6.11 hours
CNN	50/707	50	**38.23%**	4.056	19 hours
DenseNet	50/707	50	22.73%	4.04	1.5 days
ResNet	707/707	25 (running)	37.25%	3.99638	3.35 days
ResNext	707/707	3	28.75%	4.50174	9.90 days
WideResNet	707/707	20 (running)	**41.64%**	3.50708	3.60 days

Figure 17. Results for category level 3 on 10, 50 and all batches that are evaluated on subsample taken from the whole data set

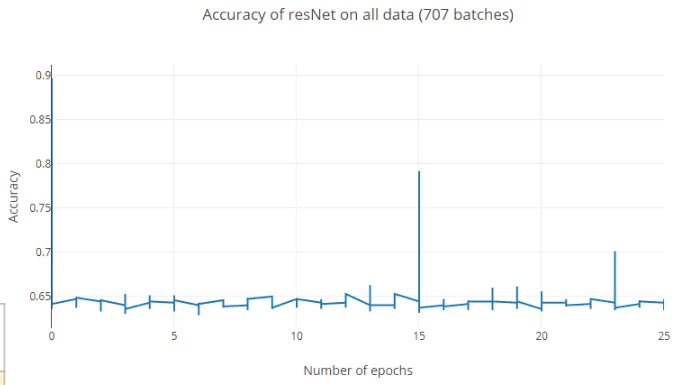


Figure 18. ResNet error per epoch

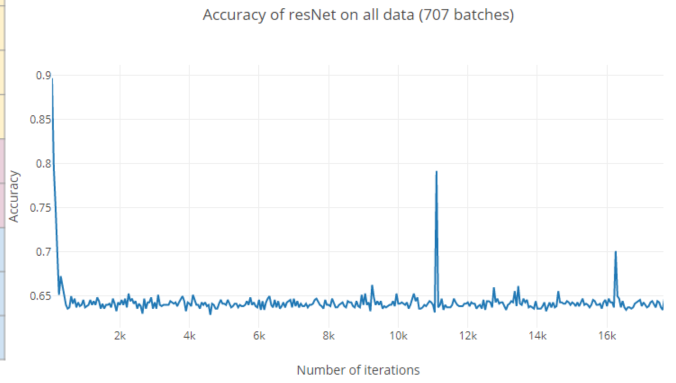


Figure 19. Residual network error per iteration

tem.

Jobs run in a virtual environment, and each user can run max two jobs in parallel in case there are free GPUs. HPC is used by whole Temple University. Each job can last for maximum 48 hours and stability of system is not guaranteed, so the program has to save model often enough in order not to lose processor's work. On the other side, writing often takes time, so in those experiments, the model was saved after every 50 batches or at the end of epoch. Models were developed on CPUs with 64GB RAM and Windows operating system and tested locally on CIFAR-10 dataset to ensure accuracy of a model.

Because of the huge dataset and limited time, the first analysis was done using CNN with 10 out of 707 batches and training separately for each category level to understand training time and possible accuracy. The training lasted for ten epochs, and accuracy, error, and time used are given in Figure 16. Accuracy for category 1 is highest, 52%, accuracy for category 2 is lower, 47%, while accuracy for third category level is the only 32%. Accuracy was measured on 1% of random samples from those ten batches and not on the cross-validation dataset described above.

The time required in those cases suggested that other models have to be as light as possible in a number of parameters. However, that wasn't enough, so many other code improvements were made to decrease the time required to train one epoch, including minimization of reading and writing time and minimization of calculations that are not required for the final result.

After all improvements, some networks were still slower, such as CNN, denseNet and resNext, so it was clear that comparison between them on the whole dataset and a huge number of epochs for category level three is not possible. Therefore, the first experiment was done on 50 randomly chosen batches with ten epochs, and the best result was achieved by the wide residual network, followed by residual network and by CNN.

CNN and dense network were also tested on 50 randomly chosen batches with 50 epochs, and their results improved slightly. The other three networks were tested on the whole dataset, and their accuracy first dropped comparing to results on 50 batches only, but as the number of epochs is increasing, results are getting better, however, when models performance is compared, the order stays the same as in case of 10 epochs on 50 batches only. The denseNet and resNext have the worst performance even though they should have the best results.

Results of resNet and wideResNet per epoch are given

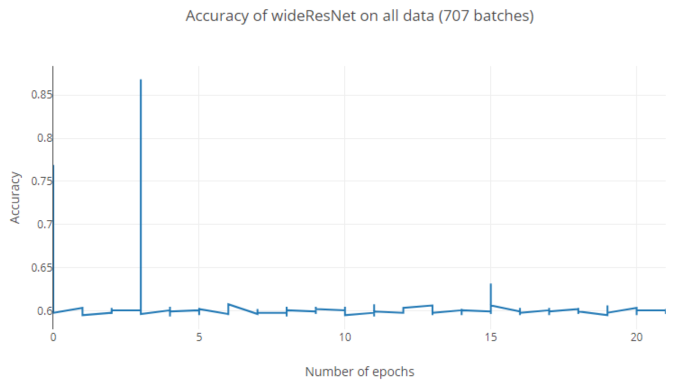


Figure 20. Wide residual network error per epoch

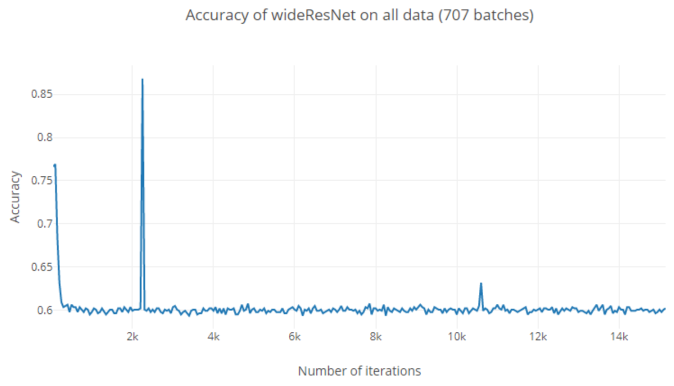


Figure 21. Wide residual network error per iteration

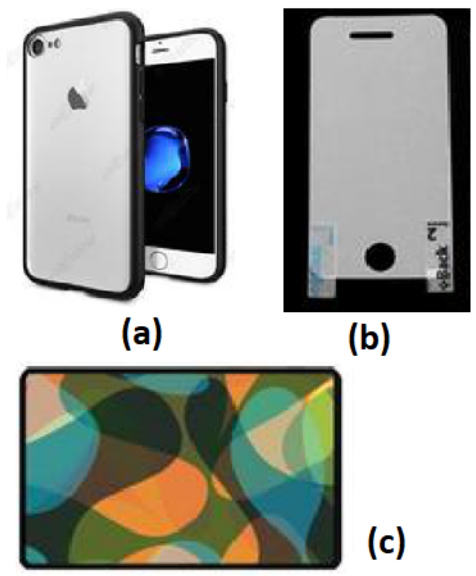


Figure 22. Error in product labeling

Item	Incorrect label	Correct label
a	Mobile	Mobile case
b	Mobile case	Mobile film protector
c	Mobile cover bag	Laptop cover bag

Table 5. Summary of data sets

on Figures 18 and 20, respectively, while Figures 19 and 21 give error per number of iterations. If those graphs are compared with the error rate from original papers per epochs number, it is visible that models haven't yet reached 30 or 60 epochs that are required for a huge drop in error (according to original papers). Additionally, the first flat part of error graphs in original papers is achieved more gradually than in the case of proposed methods, but this may be due to different scale of error axes. Otherwise, graphs seem to follow a similar pattern.

For all proposed methods error of the first flat part is around 0.6-0.65, which is more than in original papers, where it is around 0.5. This is probably due to the size of dataset, ie. number of labels which is 5027 in this problem versus around 100 in original results. Graphs of error per iterations show that results of resNet are more variable than results of wideResNet.

7. Discussion

It is not clear why resNext perform so badly, but probably because layers are very thin, so the network can't achieve it's full potential. Because of limited resources, a huge amount of time that this network requires comparing to resNet and wideResNet and bad performance, after three epochs on full dataset this model was stopped. According to results from paper, it should have a similar or best performance to resNet in this early phase, which is not happening.

DenseNet obviously can't give good results being this small. It requires wider layers, a bigger number of layers in the block, and more blocks to achieve its full potential, so the decision was not to run it on the whole dataset. Additionally, it is the most expensive method time-wise.

Because of available resources, this project requires smaller and thinner models. This would be a requirement of any smaller real system, so these results show how resistant state-of-art methods are in this case. While wide residual networks and residual networks gave similar results and were able to train acceptably fast, dense network and aggregated residual transformation (resNext) didn't pass this test. They require large memory and a lot of computational/time resources to achieve their best performances.

The dataset contains a huge variety of products, and pictures are taken from different users and are not standardized, so finding unique features of each product is challenging due to background occlusion, intra-class variability, different lighting, and viewpoints. The huge number of classes that are very similar between themselves brings an additional challenge to the algorithms.

Since the dataset contains almost five thousand different types of products, it is possible that some products have very common features. Hence, it would be difficult for a learning model to distinguish the labels. However, it is an important matter to investigate how human-level performance works on such product items.

For this purpose, a set of product items is selected randomly that are labeled incorrectly by our proposed models. For example, three items are shown in Figure 22 that are incorrectly labeled through our proposed model. The correct and incorrect labels of the three items are given in Table 5. Now, if a human tries to identify the products from the figures, it would also be challenging to him to annotate them properly. For example, item (c) in Figure 22 looks like a cover bag, but it is difficult to guess it as a mobile or laptop cover bag because each image has the same scale in the dataset. Hence, it becomes difficult to predict the cover bag for laptops or mobile. The same dilemma happens to predict the mobile film protectors and mobile cases from the Figure 22.

8. Conclusion

In this project, extensive experiments are conducted to find the best deep convolution model for the multi-class large dataset with limited computational and memory resources. Many errors and trial methods have been performed to handle the large dataset and discover a thinner and smaller network by modifying existing models that can perform well in multi-class image classification problems. After testing different existing models such as dense network, residual network, aggregated residual transformation, and wide residual network, we found the last one is the best for solving the problem with our limited resources.

Since the dataset contains multiple labels, we have a plan to train the network on one first-level category and to use this as a pre-trained network for training on the second-level category, which will be used as a pre-trained network for the third-level category. Another idea is to have multiple output objectives, one for each category and multi-task optimization. In our future work, we plan to test those ideas on the part of the ImageNet dataset [11], which also has multiple level categories for different images. This should be done by extending original state-of-the-art methods for the multi-task problem instead of considering thinner network.

9. Acknowledgment

This research work is used as a final course project of CIS 5543 (Computer Vision) course, and we are thankful to our course instructor, Dr. Haibin Ling, for his guidelines. This research was supported in part by the National Science Foundation through major research instrumentation grant number 1625061.

References

[1] Cdiscount e-commerce site. https://www.cdiscount.com/.

[2] Cdiscount's Image Classification Challenge. https://www.kaggle.com/c/cdiscount-image-classification-challenge.

[3] Image Classification. http://cs231n.github.io/classification/.

[4] Processing BSON Files. https://www.kaggle.com/inversion/processing-bson-files.

[5] D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[6] K. Fukushima. Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[9] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[12] G. Pleiss, D. Chen, G. Huang, T. Li, L. van der Maaten, and K. Q. Weinberger. Memory-efficient implementation of densenets. *CoRR*, abs/1707.06990, 2017.

[13] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv: 1409.1556v6 [cs.CV]*, 2015.

[14] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *JMLR*, 2014.

[15] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training Very Deep Networks. *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[16] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. Aggregated Residual Transformations for Deep Neural Networks. *arXiv preprint arXiv:1611.05431v2 [cs.CV]*, 2017.

[17] Y. B. Y. LeCun, L. Bottou and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] S. Zagoruyko and N. Komodakis. Wide Residual Networks. *arXiv preprint arXiv: 1605.07146v4 [cs.CV]*, 2017.