CDiscount Large-Scale Deep Learning Image Classification

Yuya Ong, Tejas Shahpuri, Naveen Muthumanickam
DS 310: Machine Learning

Introduction

CDiscount is a France based online retail platform, similar to Amazon, which has produced revenue in over 3 billion euros in the previous year. Their online product expects its catalogue of product to have over 30 billion products within the next year. For this, they require a scalable system which helps to automate their process of organizing their products into various categories. In previous iterations of their system, the company utilized a word embedding based method to automatically classify and categorize their products. However, the company decided that they would make use of the image based features they have from their product database, which they hope can increase their accuracy and precision for better classifying the product categories.

In our project we outline our modeling process as well as some of the key technical challenges we have had in this competition. Notably the challenges faced in this competition was primarily due to the immense scale of the dataset as well as the number of parameters that had to be learned in order for the model to best learn the features and classify the images. In this paper we outline some of our key observations in the dataset as well as the modeling strategies we have taken to achieve a score within the top 100 at the time of writing this report.

Challenge Objective

The overall objective of the challenge given by CDiscount is where given a set of images of a product, we are to predict the product category. More concretely, we can view this simply as a multiclass image classification problem, where the input of our mode will comprise of images and the output will be the corresponding category label provided to us. The objective of the competition is to build a model such that we can maximize the total correct number of percentages of products we get correct given a dataset.

Although the given challenge may seem very easy initially, the key challenge in this competition is based on the fact that we are given a dataset with approximately 9 million products and approximately at least more than 5,000 different classes to classify the products into. To place this competition into perspective with some of the other state of the art problem, such as ImageNet, contains 1.2 million images and only has 1,000 different classes to classify [1]. Therefore, we foresee that the largest challenge in this competition boils down to how well we can strategically scale these models across a large set of features and outputs and pushes the boundaries for the state-of-the-art regarding image classification problems.

Dataset

The dataset provided by the CDiscount competition is based on a direct database dump of their MongoDB most likely pulled from their production site. The dataset contains two sets, their training set which contains approximately 7 million of their products, and their test set which contains only 1 million of their images. Each product from their database is based on a dictionary based data structure containing the product id, up to 4 to 5 images of the product (with each image having a 180 by 180 dimension) and the corresponding category id of the product. For the test set, the structure is very similar, without the inclusion of the corresponding category id - which is our task of generating through the model we build. The training set is a bson file, approximately 65 GB compressed and around 0.5+ TB uncompressed and their testing set is another bson file as well. The data they have provided us only runs on 30% of their entire dataset, while the rest of their data will be used in a later phase for generating predictions. Based on this, we anticipate that the dynamics of the competition will highly shift and must carefully tread to ensure our model will not overfit entirely.

In addition to the testing and training dataset files they have provided, they have also provided us with a csv file containing a list of product categories with each of the categories having three different columns: Level 1, 2, and 3. The product categories forms a tree based structure, which to some extent allows us to potentially try a tree based method for the modeling process. Upon further inspection of the dataset's category, we find that there are 49 unique categories in the first level. For our second level analysis, we performed an analysis of the number of categories within the 49 unique categories in identifying the various basic statistical values of the categories. We found that certain categories from level 1 contained a minimum of 1 and a maximum of 40 sub categories and have an average of around 9 categories. Although the hierarchy information is not entirely useful out of the

blue, there may be some structures we can make use of to improve the overall precision of the model significantly as we will later describe in our modeling process.

Technical Challenges

Although the problem at first hand may seem very easy, being a multiclass image classification problem, the main challenges of the competition is based on the sheer scale of the dataset and the number of parameters that needs to be learned by the model. Therefore, the problem boils down strategically in how we can make the best use of our time and resources to effectively build a model with these given constraints to successfully build a large scale image classification model. The issue of our modeling process becomes more of an engineering based problem more so than a statistical or machine learning related problem domain that we would need to be significantly careful of.

The first challenge is based on the fact that we have a large number of data to work with, which inhibits the number of data we can load into memory when we typically train the model. Normally, when working with a fairly decent size of data, we are able to fit the data into memory - which allows us to iterate through the entire dataset in a single passthrough or epoch of training. However, in most cases of deep learning this is not entirely possible and therefore forces us to utilize a method to train the model by using a batch learning approach. The batch learning process takes a small subset of the dataset, which is typically set up as a hyperparameter for the training process, and repeats that until it has iterated through every single piece of the data - which amounts to a single epoch training. The benefit for performing this type of method allows us to train the model on a machine with a relatively low memory footprint and allows us to train the model more quickly. However, the only disadvantage to this method is the quality of the gradients when computed through the process of backpropagation will not produce the most optimal quality. This factors in strongly based on the number of batch sizes we utilize, which strongly determines the convergence rate of the model and ultimately determines the behavior of the model to find the best local optima of the model.

The next challenge that we would have to consider is the method of validating and checking the model performance when building our model. Typically in a machine learning model, we are able to perform multiple sets of experiments on the model by permuting different subsets of the training and validation set through the process of cross validation. However, since the entire modeling process as a whole takes nearly 5+ days to train, performing cross validation simply will not be possible due to the time constraints on the competition. Thus, the best we can simply do to mitigate this issue is to pre-select the data we want in our training set to use for training and validation to perform the overall training process to evaluate the expected performance of our model. We anticipated that given the size of the dataset and the number of parameters, we can most accurately describe the likely score based on using the train score as our upper bound and the validation as our lower bound score when estimating the likely leaderboard score determined by the Kaggle scoring server.

Modeling Strategies

In this section of the paper, we outline some of the various modeling strategies we have attempted and thought of during the course of the project. We have implemented only a few due to time constraints, but will list some of the key ideas and rationales we had in the methods we proposed when working on this competition.

In the previous sections, we have mentioned the use of the inherent categorical tree structure presented by the category csv files from the dataset. One of our initial strategies, prior to having access to the cluster resources, was building many smaller models which would attempt to make use of this inherent structure of the labels to effectively reduce the parameter size by building many smaller models. The structure would be similar to a tree based method, where each split between the children would be based on simpler models which uses many weak classifiers which utilizes a non-Deep Learning based approach with manual feature extraction methods to discriminate the products. Although we found this method to work to some degree in terms of scalability, we found that this method would produce many smaller models which would be very hard to maintain and develop if we were to fully implement this.

The second key strategy we have looked into was utilizing pre-trained weights, as opposed to training a model from scratch. This helps in regards to the weight initialization process as the weights that are trained from scratch usually are based on randomly initialized values - which usually do not produce the best results in terms of predictions. We can utilize the notion of transfer learning, where we take a model which was trained on a different dataset and use their learned weights on a different problem domain. The advantages of this methods allows us to train a model with relatively fewer epochs as we only need to prune the weights based on the particular dataset we have. In particular, most open architectures for existing models provide pre-trained weights based on the ImageNet dataset which allows us to take the weights and modify the architecture, by changing the output

topology of the architecture. Furthermore, another consideration to make when training the weights is to partially train the network by locking certain portions of the network while training the network to train specific areas of the feature extraction process. During this particular process we will also have to strongly consider the learning rate, as a very significantly low value due to the potential proximity we are training the model at near the local optima. The only drawback to this method would be the potential of missing other local optimums due to the way we have restricted ourselves within the domain of the ImageNet features.

A third strategy we have looked into, based on the extension of the previous strategy, is to produce a ensemble of pretrained models to combine the results of the prediction. By utilizing various weight topologies and different learning methods, we can have our model to account for the variance in the different learned features from these different models. Notably two different approaches for ensemble methods in the literature we can make use of are based on voting and averaging methods. For voting methods, there exist two methods: plurality voting and weighted voting. The strategy for using a weighted voting scheme allows us to account for the variability in the modeling process and use the different learned features to improve the overall accuracy of the challenge objective.

Another strategy we employed in our modeling process was to make use of data augmentation methods to increase the variability within our dataset. We introduce data augmentation in our pipeline as we noticed a huge number of variability as well as imbalance in our dataset. To increase the number of samples, we can use basic image transformations and perturbations within the dataset. Types of perturbations include rotation, shearing, reflection, translation, random cropping, and resizing. However, when performing these types of operations, we would have to be very wary of the amount of perturbations we introduce as this may ultimately affect the parameters learned in the process. For this, we would have to closely investigate how much randomized perturbations we want to introduce in order to measure the relative gains we obtain in terms of the performance.

Finally, one of the last considerations to make when formulating this machine learning problem is how we formulate the probabilistic interpretation of the learning objective. For this, we can consider two different interpretations of the same problem by changing the way we formulate our output probability. In this Kaggle competition, the problem formulation is based on the notion of predicting the product category based on the given product ID, and for every given product ID, we are provided with up to 4 to 5 different images of the product. Therefore, when building our model, we can consider either a image based perspective or the model based perspective. From an image based perspective, given an image x, we can predict a probability p, where $p = Pr(Y = y \mid x)$, where Y is a category within {1, …, 5270}. Thus, given a product T, where T = $\{x_1, x_2, \ldots, x_n\}$ and n is the number of products provided to us, within T, we can consider the *average probability for each of the prediction in the set T* of our image as our general prediction for T as the probability q, where $q = (p_1 + p_2 + \ldots + p_n) / N$, where N is the total count of images within the set of T. On the other hand, we can consider another interpretation based on a product based probabilistic interpretation of the model where we jointly model the probability of all of the image. More concretely, we can define this probability as a joint probability q, where $q = Pr(Y = y \mid x_1, x_2, \ldots, x_n)$ for the set of images in the product T. The justification for formulating the problem in this fashion allows is based on the following intuition: consider a product, such as a cell phone case, where you have multiple images of the case itself. With a majority of accessory based products , often sellers provide extra add-ons such as a screen protector or a screen cloth. If we consider the object as an independent product alone, the likelihood of the image of the cloth by itself being predicted as a cell phone protection case, the ground truth label, would be likely lower and therefore bringing down the combined probability of the product being considered a cell phone case because of the individual probability. However, if we consider the joint probability of the model based on the priors of the product images being together within the same context, we would be able to better model the semantic occurence of the likelihood between the two images being together - thus improving the model's predictive power in seeing that the cleaning cloth is part of the cell phone case as an additional accessory product.

Models

In this section, we discuss the models we have implemented (and also will implement) to model the problem used in the CDiscount problem. For the duration of the project, mostly due to time constraints and resource allocation, we were able to fully train only a single model - the Inception V3 model. However, we will make the best use of the remaining allocated time with the resource to train another model, or the Xception model.

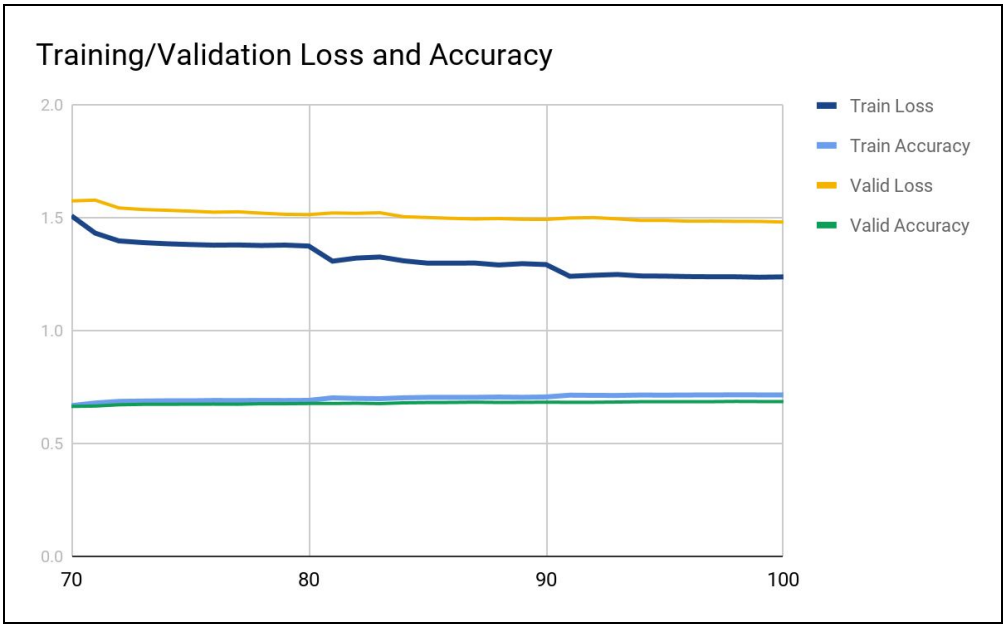
The first model we trained was based on a pre-trained Inception V3 model proposed by Szegedy et. al [2]. The key advantage of this architecture over traditional convolutional neural network architectures is the fashion in which this model attempts to learn a nonlinear function of the image model, as opposed to a linear function. They achieve this through learning various feature

representations through different kernel sizes within the same level and utilizes a global average pooling to combine the different feature feature representations. This reduces the need for the fully connected dense network towards the output of the end of the model and therefore reduces the number of computations necessary for training the model. For this competition, we trained a model using the Inception V3 model based on the pretrained weights trained on the ImageNet dataset. During our training process, we first replaced the output softmax layer from the 1000 dimensional vector output of the softmax layer to a 5270 dimensional vector output. Prior to that, we then train the parameters of the output softmax based on the weights defined by the ImageNet network to formulate the activation weights of the output softmax function. This allows us to first exploit the learned lower level features from the ImageNet dataset through the process of transfer learning. During this process, we lock the weights of all other weights except the output layer so that they are not augmented during the backpropagation process. In this stage we use a relatively large batch size and a larger learning rate to help the model to converge towards the local optimal relatively quickly. In the next phase of training, we unlock the weights for all other layers above the output softmax function and prune the weights based on the feature of the dataset. This process will allow us to improve the overall precision of the model by fine tuning the learned features based on the weights learned from the CDiscount dataset. In this process we use a relatively small batch size and a small learning rate to ensure that the gradients will not cause the model to diverge from the local minima.

The second model we plan to train is based on the Xception (Extreme Inception) model, which is based on an improved version of the Xception V3 model, proposed by Francois Chollet [3]. The model improves on the original Inception's notion of the hypothesis where the mapping of cross-channel correlations and spatial correlations in the feature maps of convolution neural networks are entirely independent. The model makes use of depthwise convolutional neural networks followed by a point wise convolution operation which helps to further make better use of the computational resources during parameter optimizations. We plan to follow a very similar training methodology mentioned in previously.

Results

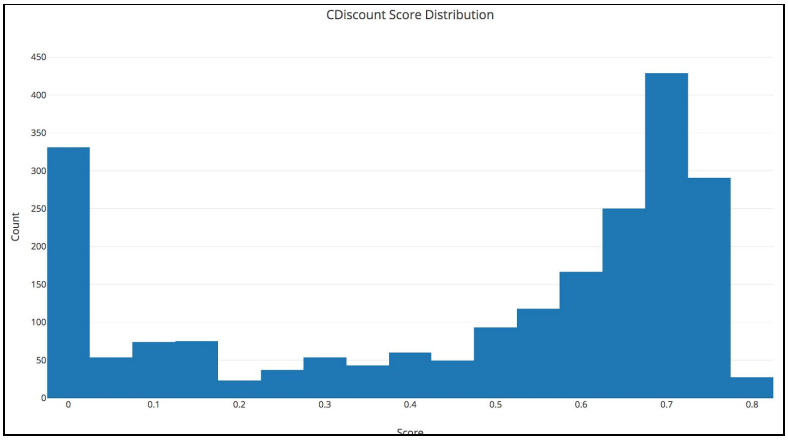
We trained the Inception on a single node from a GPU cluster with 4 Nvidia P100 GPUs, which took approximately 5 days for training and 3 hours for inference across our test data. The training and validation loss and accuracy of the model's last 30 epochs is shown in the graph below. During our training process, we anticipated that our model's leaderboard score can be predicted as by the training accuracy being part of our upper bound while our lower bound can be bounded by our validation accuracy.



As of posting results onto the Kaggle submission server, our results based on December 6th, 2017, we have placed 96th place - within the top 100 position of the leaderboard and on the 16th percentile of submissions (this may change with the additional submissions in the future). Below we also present a table of our previous submissions to the Kaggle leaderboard along with the

scores we have obtained (top) and the distribution of the overall score for comparison against other submissions as of December 7th (bottom).

Date	Model	LB Score
11/04/2017	Sample Baseline	0.00868
11/11/2017	CNN Model 1	0.16718
11/20/2017	CNN Model 2	0.00892
11/20/2017	CNN Model 3	0.00868
11/20/2017	Inception V3 Prototype Model	0.02502
12/4/2017	**Inception V3 Final**	**0.70146**



Analysis and Discussion

As a result of this competition, we have been able to successfully implement a large scale image classification model which can classify different products. Throughout this competition, we have taken various experiences from some of the kernel and discussions presented within Kaggle as well as invoke some of our own ideas into practice to figure out a strategy to score relatively high within the leaderboard. However, there are some key takeaways that we have found valuable in this experience.

First, in this project, one of the major strategies is to be resourceful in how we attack our problem. In this case, the use of pre-trained weights have significantly boosted our results and have decreased our overall training time. In this case, we found that working off of the state-of-the-art results and building off of the work of others can be a valuable resource in being able to implement something relatively quickly and effectively in these types of applied areas. Another area we focused on during our model training portion was the hyperparameter selection process in our models. As we were limited in the number iterations possible for us to execute our models, due to the massive scale of our problem, we had to be very careful in how we execute our program. This forced us to be very thoughtful in choosing the hyperparameter values and required us to review the literature and the results of previous works to best determine the optimal values and strategies. Finally, one of the most valuable lessons in this competition was the experience of working under the limited hardware constraints. We found that certain limitations and constraints on the problem can force us to think creatively for out of the box solutions which can potentially lead to interesting or unique solutions - even though they may not be effective at first sight.

In short, our team has learned a lot from this experience and hope that we continue to work on this competition prior to the deadline of the project.

References

1. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 248-255). IEEE.
2. Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.
3. Chollet, François. "Xception: Deep Learning with Depthwise Separable Convolutions." *arXiv preprint arXiv:1610.02357*(2016).