Predicting Movie Genres

CS109B Advanced Topics in Data Science (Spring 2017)
Final Project Group 29
Calvin J Chiew, Tim Hagmann, Ji Hua

INTRODUCTION

In this project, our aim was to predict movie genres using movie posters and other movie metadata. The project consisted of two main parts. In the first, we trained "traditional" machine learning models using movie metadata. In the second, we developed deep learning networks using movie posters. Finally, we compared all the models based on our chosen performance metrics.

We approached the task of genre prediction as a multi-label classification problem. We chose to do this as it is difficult for a movie to fall neatly into one category – most movies are naturally classified into more than one genre. We used 19 categories as defined by The Movie Database (TMDb)[1], namely Adventure, Fantasy, Animation, Drama, Horror, Action, Comedy, History, Western, Thriller, Crime, Documentary, Science Fiction, Mystery, Music, Romance, Family, War and Foreign.

METHODS

Data Scraping

Our sources of data were The Movie Database (TMDb) and the Internet Movie Database (IMDb)[2], which were accessed using the TMDb API and IMDbPY library respectively.

We used the TMDb API to scrape information on movies that have `tmdb_id` of 1 through 160,000. About 60,000 of these `tmdb_id`'s were empty, returning about 100,000 actual records. Given that `tmdb_id`'s were not assigned in chronological order or in any particular pattern, this represented a random sample of all movies listed on TMDb. We then dropped about 30,000 records that were missing poster paths and `imdb_id`'s, leaving us with about 70,000 movies. We believe this was a sufficiently large sample size to develop our "traditional" and deep learning models.

Next, we extracted additional metadata from IMDb for this list of 70,000 movies and matched them to the TMDb records based on `imdb_id`. As there was an overwhelming amount of metadata available for each movie, we selectively chose variables to extract. We identified features that were potential predictors of movie genre based on our prior beliefs.

From TMDb, we extracted the following variables: title, plot summary, production company, release date, runtime, budget, revenue, popularity and average user rating. In addition, we downloaded the poster in w154 resolution and `.jpg` format for each movie. From IMDb, we extracted the following variables: cast (first 5 names listed), directors (first 2 names listed), writers (first 2 names listed) and US MPAA picture rating. (Information on cast, directors and writers were in the format of `personID`, which were 7-digit identifiers used by IMDb.)

Exploratory Data Analysis

We conducted a preliminary exploration of the data, visualizing how the extracted features were associated with movie genre. We also generated a heatmap of the movie genre pairs, to see which genres tended to go together.

Data Preparation

The outcome variable, movie genre, was classified into 19 categories as mentioned above. We decided against regrouping the genres because even if we had chosen to do so, we would have only collapsed a few groups based on our genre pair analysis. We did not feel that a small reduction in genre categories would make a significant difference in the larger picture. Any other "artificial" regrouping would have limited the utility of the eventual model. We applied a multi-label binarizer, i.e., one-hot encoding[3], to the genres, obtaining 19 columns of binary flags, one for each genre.

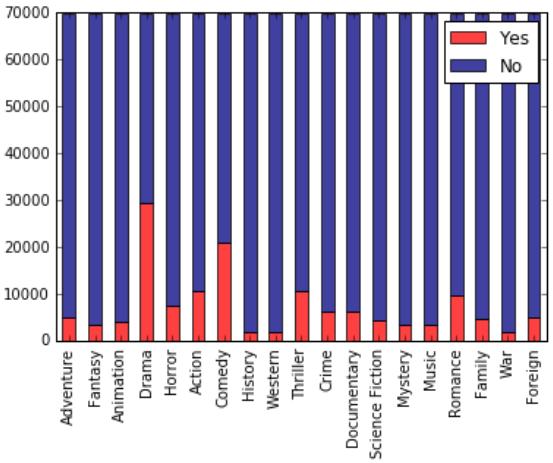

Figure 1. Class imbalance in the dataset.

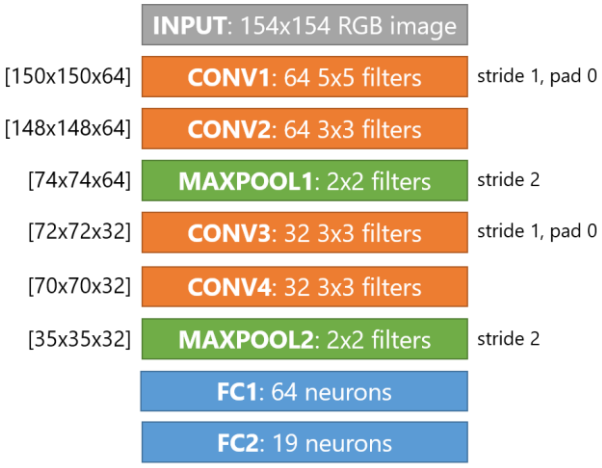

Figure 2. Architecture of CNN trained from scratch.

To simplify the problem, we did not use the genre tags on IMDb, which may have been different from TMDb for some movies.

We used bag-of-words[4] analysis to convert the title and plot summaries into features. We ignored stop words and only considered words that made up at least 1% of the corpus, which resulted in 389 words in the vocabulary. The final weighted frequency matrix was generated using Term Frequency times Inverse Document Frequency (tf-idf)[5].

As there were too many different production companies in the dataset (which would have resulted in an exceedingly large number of dummy variables), we only considered the top 20 production companies with the most films. We applied one-hot encoding to obtain columns of binary flags, one for each of these top 20 production companies. Movies that were produced by other companies would have "0" for all indicator variables.

We did the same for cast, directors and writers, taking only the top 150 cast and top 75 directors and writers with the most films. We also transformed the release date variable into release year and month, and applied one-hot encoding to categorical variables, such as US MPAA picture rating. All variables were scaled prior to modelling.

To ensure that all the posters had a standard input array size, we used the Python Imaging Library (PIL)[6] to resize all of them to 154 by 154 pixels. Then, for each pixel, we extracted 3 colour channel (RGB) values, creating matrices of dimensions 154 by 154 by 3.

"Traditional" Models

We chose to fit two "traditional" classifiers using random forest[7] and k-nearest neighbors (kNN)[8]. We used the common approach of splitting the data two-thirds for training and one-third for testing[9]. Class imbalance (visualized in Figure 1) was addressed by using class weights in the models. Given the multi-label problem, a one-vs-rest (OvR)[10] strategy was employed, which consisted of fitting one classifier per genre. For each classifier, the genre was fitted against all the other genres. This was done using the scikit-learn library[11] in Python. Parameter tuning was performed via 5-fold cross-validation[12]. We tuned the number of trees for the random forest, and value of k for kNN.

As a benchmark, we also created a baseline model, which randomly flagged a genre at a probability of around 10%, which was the overall proportion of true positive flags across all 19 classes.

Deep Learning Models

We trained a small convolutional neural network (CNN)[13] from scratch using Keras[14] with TensorFlow[15] backend, and fine-tuned another existing pre-trained network. To speed up computation, the models were run on the "p2.xlarge" GPU compute instances on Amazon Web Services (AWS)'s[16] Elastic Compute Cloud (EC2)[17]. We randomly selected 10,000 posters for training, 10,000 for validation and 10,000 for testing.

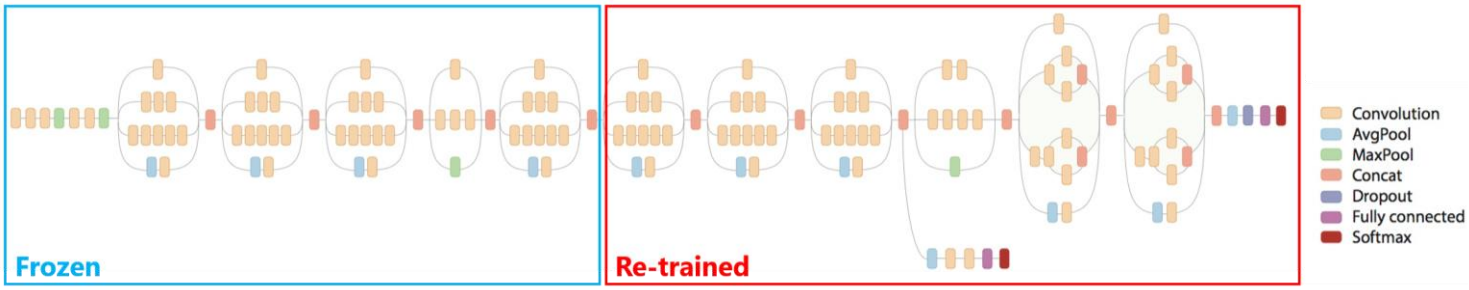

Figure 3. Architecture of CNN fine-tuned from Inception-v3.

The architecture of our CNN is shown in Figure 2. We used the 'Adam' optimizer with a binary cross-entropy loss. The activation function was rectified linear unit (ReLU)[18] for the hidden layers, and sigmoid for the output layer. We set the learning rate to be 0.1, and reduced it by a factor of 0.2 when validation loss plateaued for 1 epoch, until a minimum of 0.001. Batch size was tuned to 500 and number of epochs to 10.

We also employed L2 regularization[19] and dropout (0.25 on hidden layer, 0.5 on fully-connected layer). Weights were initialized with the Glorot uniform method[20]. The images were centered beforehand and data augmentation (re-scaling, rotation, width shift, height shift and horizontal flip) was also utilized.

For the pre-trained network, we decided to fine-tune Inception-v3[21], which was developed by Google using data from the ImageNet Challenge in 2012[22]. We chose to work with this model because of its high classification performance and its availability in TensorFlow.

In the initial phase, the base model was loaded, and a fully connected layer (with 1024 neurons) and a dense layer (with 19 output classes) were added on top. All Inception-v3 layers were frozen for this phase. After the top layers had been trained for 10 epochs, layers from the 6th inception module and above were unfrozen for further re-training for 50 epochs, while layers below were kept frozen. In contrast to the above network trained from scratch, we used stochastic gradient descent (SGD)[23] with momentum of 0.9 as the optimization method and a low learning rate of 0.0001.

The architecture and parameter settings of both networks were tuned by monitoring and visualizing the loss curves over the training process. Like the "traditional" models, class weighting was employed to address class imbalance.

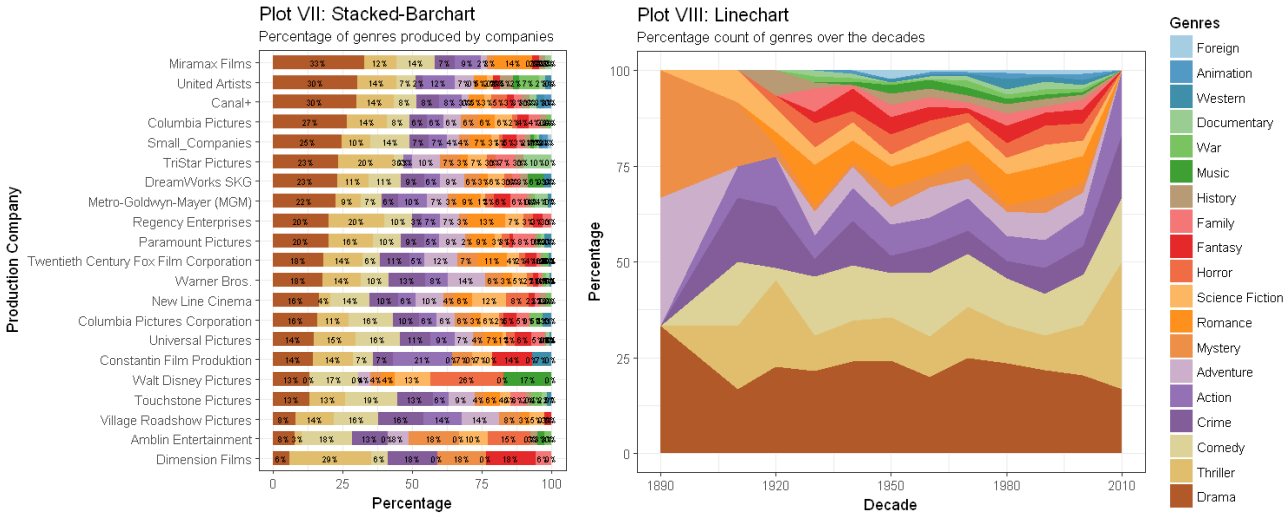

Figure 4. Selected visualisations from exploratory data analysis. Left: Production company. Right: Release decade.

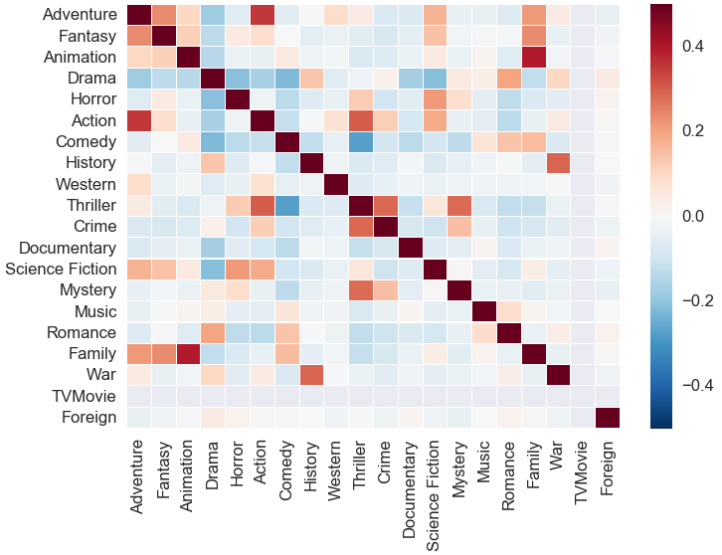

Figure 5. Heatmap of genre pairs.

Model Evaluation

Evaluation of model performance was tricky due to the multi-label nature of the task. After consideration, we decided to evaluate our models using test accuracy, precision, recall and F1 score[24]. Test accuracy was the most intuitive to understand, but had to be interpreted with caution. As most of the true labels were negative (coded "0"), a model could easily achieve high test accuracy simply by predicting all genre flags to be "0".

On the other hand, precision, recall and F1 score allowed us to focus only on the positive flags, as they do not give credit for classifying true negatives. Recall indicated the model's ability to find all the positive flags, while precision reflected the model's ability to not flag as positive a genre that is negative.

The relative contribution of precision and recall to the F1 score were equal. However, if we preferred fewer but highly spot-on genre labels, as opposed to more but less accurate labels, we might consider precision to be more important than recall. In our multi-label scenario, all performance metrics refer to a weighted average across the 19 classes.

RESULTS

Exploratory Data Analysis

Figure 4 shows selected visualisations from our exploratory data analysis. We found most variables to be associated with movie genre. For example, major production companies produce different mixes of genres. In terms of release date, the proportion of each genre appeared to stay relatively stable across all decades. However, in the 1920s there was a peak of thriller movies and less comedies. Music movies were particularly popular in the period from 1950 to 1980.

From our genre pair heatmap (Figure 5), we found the following pairs of genres to be strongly correlated: Action-Adventure, Animation-Family, Action-Thriller, History-War, Thriller-Crime and Thriller-Mystery. In contrast, these pairs of genres were negatively correlated: Drama-Comedy, Drama-Science Fiction and Comedy-Thriller.

"Traditional" Models

Both our random forest and kNN models achieved similar overall test accuracies of 90.30% and 89.06% respectively, which were higher than the baseline model (81.43%). The random forest had higher precision than the kNN model (0.814 vs. 0.374), but lower recall (0.122 vs. 0.184) and F1 score (0.160 vs. 0.224).

Deep Learning Models

Both our original and fine-tuned networks also achieved high overall test accuracies of 89.52% and 89.51% respectively. However, both networks had very low precision (0.000 and 0.140 respectively) and recall (0.000 and 0.004 respectively). It appeared that the

Model	Test Accuracy (%)	Precision	Recall	F1 score
Baseline	81.43	0.196	0.105	0.121
Random forest	90.30	**0.814**	0.122	0.160
k-Nearest neighbors	89.06	0.374	0.184	0.224
CNN (Trained from Scratch)	89.52	0.000	0.000	0.000
CNN (Alternative multi-class)	90.02	0.005	0.052	0.006
CNN (Finetuned from Inception-v3)	89.51	0.140	0.004	0.007

Table 1. Summary of model performance.

networks were simply predicting "0" for all the genre flags. Tuning the learning rate, batch size and other parameter settings (such as early stopping instead of reducing learning rate on plateau) and architecture designs did not change their performance.

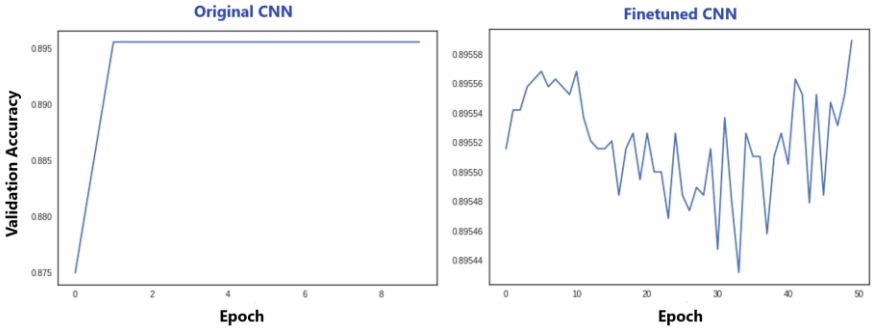

Figure 6. Validation accuracy over training epochs of both networks.

Visualizing the training processes (Figure 6), it appeared that more epochs did not improve their performance which was surprising. The largest gains were in the first few epochs.

Alternative Deep Learning Models

Given the low precision and recall scores of our networks despite extensive tuning, we attempted several alternative methods of developing our network. First, instead of training on 10,000 random samples, we selected 1000 posters belonging to each genre (total 19,000 training samples), and simplified the multi-label problem to a multi-class one. Doing this did not improve the precision and recall scores.

Second, for each genre, we selected 1000 posters that belonged to that genre, and 1000 posters that did not belong in that genre, and trained a network to predict "Yes" or "No" for that genre label. Again, the precision and recall did not improve. We examined the predicted probabilities from the network and found that prediction was equivocal (predicted probabilities close to 0.5) for almost all test samples. This was the case even for what we intuitively felt were "easier" genres to predict such as Animation and Horror.

Third, we went back to our original setup using class weights, but attempted to determine an optimal predicted probability threshold for each genre, beyond which we might label a genre positive. We tried a wide range of finely-spaced candidate threshold probabilities, iterating through each to select the one that gave the optimal F1 score, for each genre. However, the best threshold selected was always the smallest value in the candidate range. Again, we examined the predicted probabilities from the network, and found the variance of predicted probabilities for each genre across test samples was close to 0. This meant that regardless of the test image inputted, the predicted probabilities all turn out the same! In the discussion below, we explore possible explanations and solutions to this conundrum.

DISCUSSION

Our deep learning networks performed poorly in terms of precision and recall. This was despite numerous tuning efforts and trying alternative methods of sampling and defining the task as described above. The issue was unlikely due to class imbalance as all methods we used addressed it one way or another. It was also unlikely due to overfitting because otherwise our first or second alternative method would have worked.

One possible explanation is that the task of predicting genres using posters is truly very difficult. In a crude experiment, we looked at some of the posters ourselves and found that we could not guess the genres correctly most of the time. A lot of the posters were simply made up of actors' faces, with very little other (non-textual) information to base a prediction on. As works of art, some posters were also very "creative" and not typical of what one would expect from a movie of that genre. Finally, some of the genres, such as Adventure, have very broad definitions and so a wide range of movies with different styles (and poster features) can still be classified as Adventure.

A possible solution would be to "cherry pick" posters that we think are representative of each genre, although this would introduce some subjectivity into what we consider as "representative". This would perhaps alleviate the problem of a low correlation between posters and genre.

Another possible explanation would be that we had too many classes, although this seemed unlikely given that our second alternative method, which reduced the problem to a single genre only, did not improve network performance. A possible solution (if this was the issue) would be to cluster all the genres into only a few categories and approach it as a multi-class problem with fewer classes.

There were also several weaknesses unrelated to the deep learning part of the project. In our "traditional" models, we ended up with a large number of predictor variables (almost 880 including dummy variables). We could have applied PCA prior to modelling for dimensionality reduction. We could have also explored other possible models such as support vector machine and boosting.

An alternative, and perhaps fairer approach to using only the genre tags from TMDb, is to match the tags from both TMDb and IMDb, and consider the intersection of the two sets as the true labels.

Despite all these limitations, we believe our project had several merits. We had a large sample size and a wide variety of predictors which were scraped from both IMDb and TMDb, to supplement the posters. We used methods that were appropriate for the multi-label problem and accounted for class imbalance.

In conclusion, our best genre prediction model was the random forest, which obtained a precision of 0.814 and recall of 0.122. Figure 7 shows the breakdown of its performance across the 19 genres. Genres with above-average precision and recall such as Animation and Drama were easier to predict. The random forest was trained on movie metadata. Attempts to train deep learning networks for the same task using movie posters did not turn out useful.

To view our iPython notebooks and raw data files, please visit our GitHub repository at https://github.com/greenore/deep-learning-project. To view a summary of our findings in a short screencast, please visit https://www.youtube.com/watch?v=oyXUTy3y2-k.

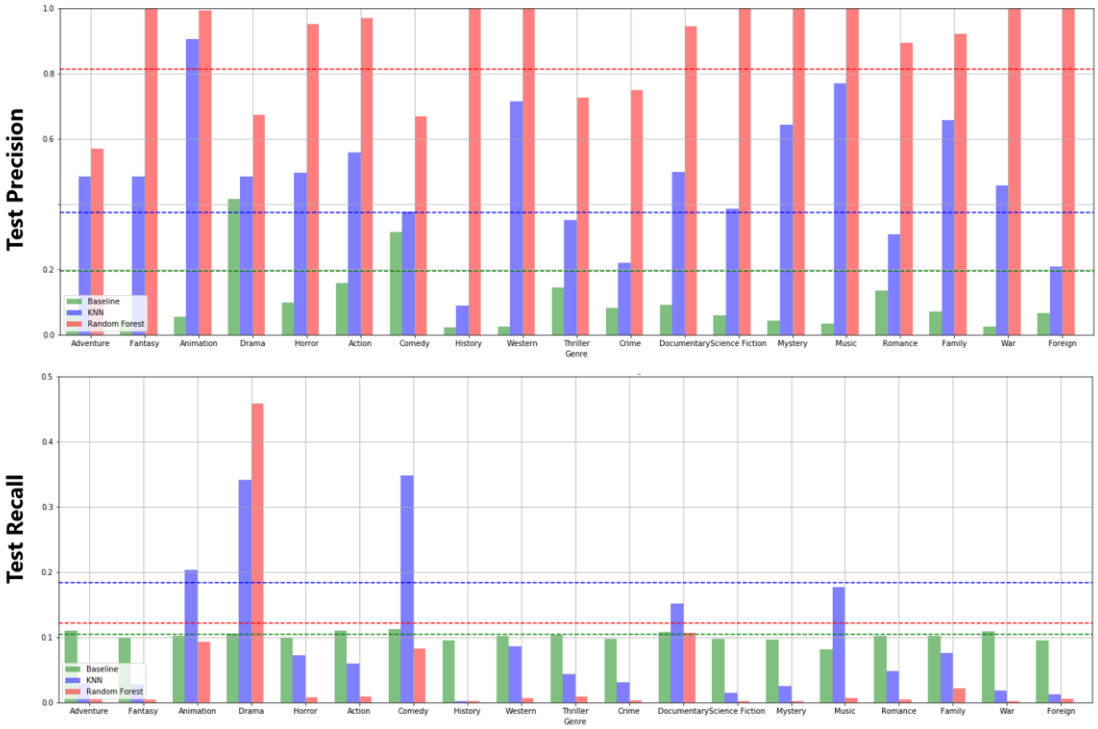

Figure 7. Precision and recall of traditional models by genre.

REFERENCES

1 https://www.themoviedb.org
2 http://www.imdb.com/
3 Hastie, et.al., (2009), Introduction into statistical learning, pp 85-87
4 Sivic, (2009), Efficient visual search of videos cast as text retrieval. pp. 591–605.
5 Rajaraman, (2011), Mining of Massive Datasets, pp. 1–17.
6 http://www.pythonware.com/products/pil/
7 Hastie, et.al., (2009), Introduction into statistical learning, pp 319-321.
8 Hastie, et.al., (2009), Introduction into statistical learning, pp.39-42.
9 Hastie, et.al., (2009), Introduction into statistical learning, pp.176-177.
10 Tsoumakas, (2007), Multi-label classification: an overview, pp. 1-13.
11 http://scikit-learn.org/
12 Hastie, et.al., (2009), Introduction into statistical learning, pp.181-183.
13 Collobert et.al., (2008). Deep Neural Networks, pp. 160-167
14 https://keras.io/
15 https://www.tensorflow.org/
16 https://aws.amazon.com/
17 https://aws.amazon.com/en/ec2/
18 Hahnloser, et. al, (2000), Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit, pp. 947–951.
19 Rosasco, et. al. (2015), A Regularization Tour of Machine Learning.
20 Glorot, et. al., (2010), Understanding the difficulty of training deep feedforward neural networks.
21 Szegedy, et. al. (2015) Rethinking the Inception Architecture for Computer Vision.
22 http://www.image-net.org/challenges/LSVRC/2012/
23 Bottou, (1998), Online Algorithms and Stochastic Approximations.
24 Hastie, et.al., (2009), Introduction into statistical learning, pp.147-148.