CS230

Learning TV Ads Video Image Frames Products Classification

Raul Salles de Padua
SCPD Student - AI Graduate Certificate
Stanford University
rpadua@stanford.edu
https://youtu.be/MB1ngZ22kiU

Abstract

Aiming to build a seed algorithm to classify single image frames teared down from Brazilian TV advertises. This project tackled a private data set of TV ads videos from Shopping Brasil (SB)[1], a company that provides consumer products offerings monitoring information by scrapping information from several digital and non digital merchants in Brazil. The company currently has high overhead costs associated to providing this information to its customers. Prior associated work was performed within Stanford community by Andrej Karpathy applying Convolutional Neural Networks (CNNs) to video classification[3]. I pursued one of the approaches taken by Karpathy et al, by classifying single image frames from videos (stacks of images). To add complexity, the data presented a high class imbalance towards negative image frames (no product displayed in the frame). I have generated more than 10,000 1s single frames, building a CNN that reached accuracy metrics of 78% and 82% in validation and test sets respectively. Benchmark architectures were adapted to this task, Single Frames Classification (by Karpathy et al) and VGG16[9]. Their performances in the validation set were 76.67% and 52.93% respectively.

1 Problem Statement and Related Work

I endeavored to build a seed learning algorithm/ diagnosis to classify products shown in TV advertises in order to help Shopping Brasil (SB)[1], a company that provides consumer products offerings monitoring information by scrapping from several digital and non digital merchants' ads in Brazil. SB provided real world TV ads to this project to at least automate partially its current undergoing processes of product and price TV ads' monitoring and reporting. Such processes are currently executed entirely manually with more than 10 people watching every day ads overnight to provide SB's customers recent current day minus one information. This human performance level has 95-98% accuracy including headcount to check manual insertions by sampling, resulting in additional costs not shared by the company. I tackled data provided of 3 of the most advertised products in Brazilian TV: samsung mobile phones, other mobile phone brands and beers (aggregated).

Distinguished endeavor was undertaken within Stanford community. It was conducted by Andrej Karpathy, applying Convolutional Neural Networks (CNNs) to video classification[2]. Karpathy et al identified several challenges applying CNNs to this level of application, with one being no benchmarks that can match the variety and magnitude of existing image data sets because videos are significantly arduous to collect, expensive to store and to comment. The same challenges and constraints were faced pursuing this project, as SB could only share data from the last 6 months

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

of TV ads (July - December 2020). I have trained an adapted model from Karpathy et al[2] used to implement single frames classification on the data to use as benchmark (see all performance comparisons among classes in Table 1).

2 Dataset

2.1 Overview

SB provided around 500 videos the most TV advertised product categories in Brazilian open TV from July to December 2020: samsung mobile phone, other mobile phone brands and beers. Another class of frames is "not phone beer", which means none of the product categories, or negative.
The input to the image classification algorithms are frames, taken from every second of each video file (each a stack of images) using FFMPEG inside Terminal[4]. Classes are highly imbalanced towards negative cases as shown in Figure 1. I tackled such issue by weighting each class by the inverses of their percentage distribution in the training set.

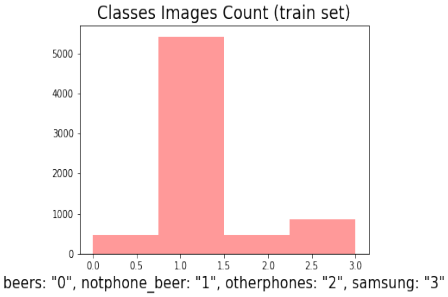

Figure 1: Classes imbalanced distribution towards negatives

2.2 Image frames generation, labeling, and preprocessing (including augmentation techniques)

We processed, provided under courtesy, raw video .mp4 files following the steps below:

1. Frames generation from the video files. Such framing process is performed applying FFMPEG[4]. In terminal, I executed the following command line to create a single <output>.png frame automatically generated in the desired directory:

   ```
   ffmpeg -i video_file.mp4 -vf fps=1 img/output%05d.png
   ```

   Examples of image frames are displayed in Figure 2.
2. Folder structuring and labeling of of all images in the 4 classes using keras preprocessing and other python data science libraries
3. Training, Development and Test sets split was performed with roughly 70, 15 and 15 percent image frames distributions respectively
4. Data augmentation on the training set by applying zooming, horizontal flipping and rotation

Images' resolution were kept the same as the video files stored: (192, 240), RGB. An example of a TV ad can be seen in an open access web link (courtesy of Shopping Brasil)[5].

3 Model Building and Hyperparmeter Tuning

3.1 Baseline

In order to evaluate which architectures would work in the data, I have trained a "Shallow CNN", inspired by Tersorflow.org tutorial[10], to perform initial analysis on results and to diagnose the data

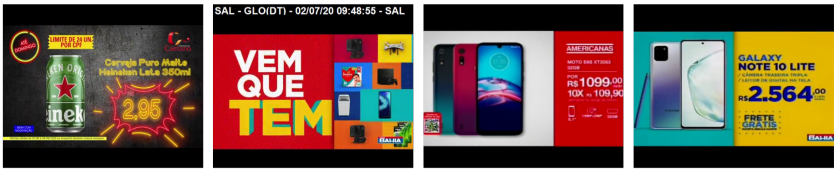

Figure 2: Image frames teared a part from merchants' ads

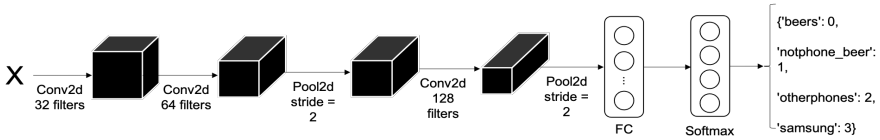

Figure 3: "Shallow CNN" trained to baseline image frames TV Ads classification

and define next steps going deeper. Its high level architecture illustration is displayed in Figure 3. To be consistent, I have trained all iterations using a batch size of 64 with 350 epochs and learning rate α to be 0.0001. Figure 4 illustrates diagnostic baseline model training history. Validation loss reached in the 'Dev Set' was 0.8979 with an accuracy of 67.58%; evaluating in the 'Test Set' loss minimized to 0.7393 and accuracy finished at 71.97%.

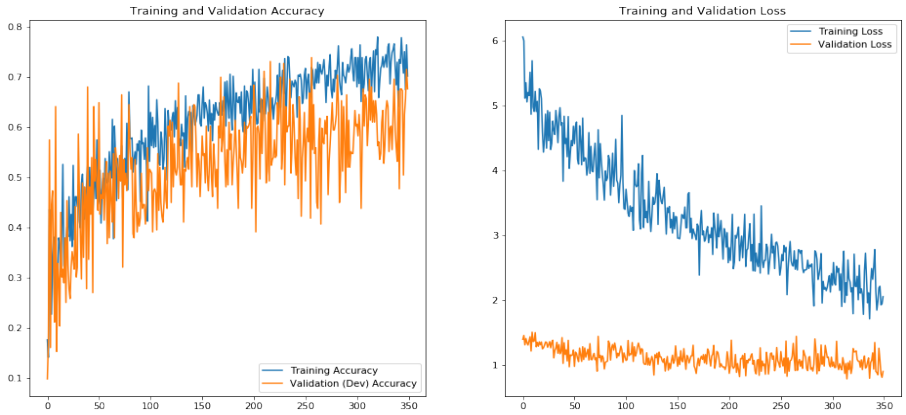

Figure 4: Training of Shallow Neural Network as diagnostic baseline

3.2 Deeper Models Architectures and Tuning

In efforts to tackle both bias and variance problems faced during diagnosis, I increased complexity in the architecture illustrated in Figure 3. More convolutions as well as fully connected (FC) layers were included in the final architecture of this project. After running several iterations, I have reached final architecture illustrated in Figure 5. Inputs shape is (192, 240, 3), kernel sizes of 5 and 3 with stride of 1 and same padding, and max pooling of 2 with stride of 2.

Before arriving to this final architecture design I ran an intermediate deep model, where the first hidden layer had 32 units rather than 64 and kernel of 3, and without the second Conv2d-256, the last Conv2d-512 layers, as well as only one 64 unit FC layer before the softmax output. It performed better, with 76.95% accuracy and loss of 0.7695 in the 'Dev Set'.

Those improvements pointed towards training the architecture illustrated in Figure 5, 'Dev Set' accuracy increased to 77.92% with loss of 0.7263. Results in the 'Test Set' performed even better with accuracy of 82.17% and loss of 0.4976. Figure 6 displays accuracy and loss training history.

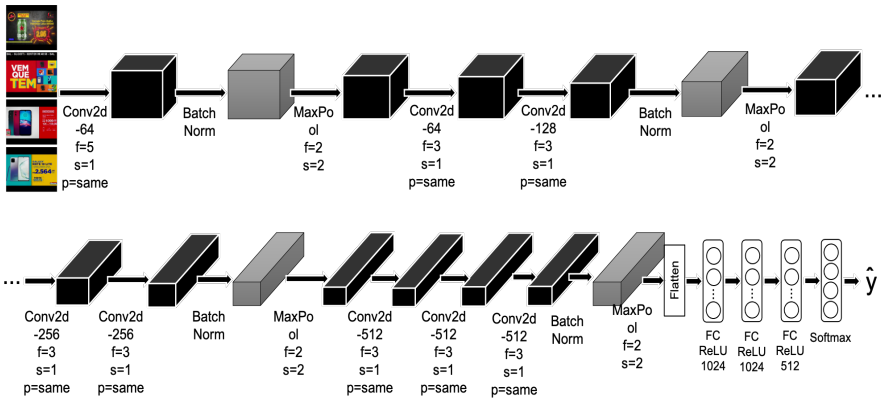

Figure 5: "Deeper Neural Network" architecture (inspired by publicly available CNN VGG16 with a principle of doubling "C" on deeper layers)

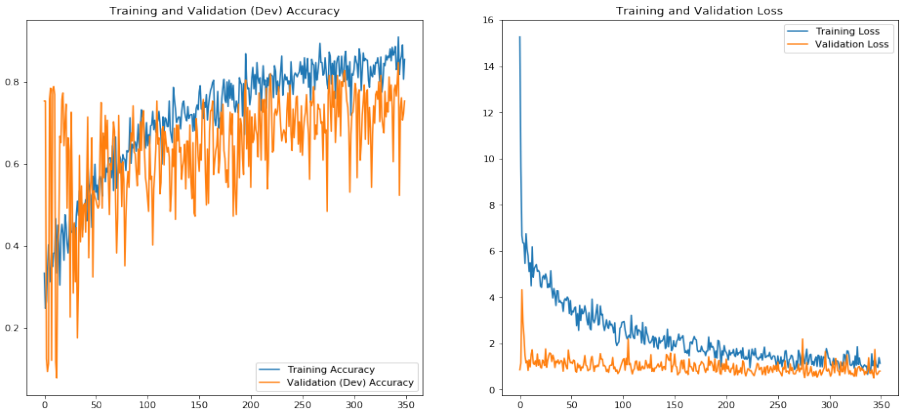

Figure 6: "Deeper Neural Network" training history

I used as benchmarks an adapted Karpathy's paper baseline model to single frames, training 100% to this task, and classic VGG16, adapting final FC and output layers. I reached accuracy in the 'Dev Set' of 77.34% and 52.93% respectively. History of these training iterations are shown in Figure 7.

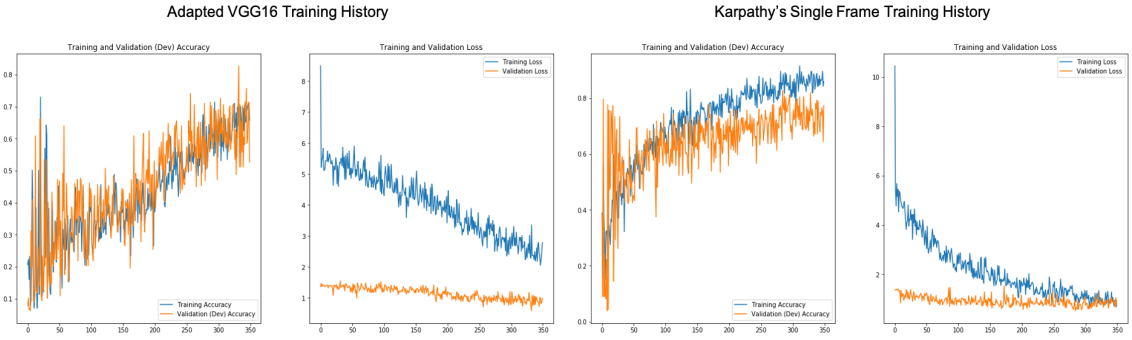

Figure 7: Training history on benchmarks

Classes-Metric	Shallow Model (Dev)	Shallow Model (Test)	Final Deep Model (Dev)	Final Deep Model (Test)	VGG16 (Dev)	Karpathy's single frames (Dev)
Beer – Precision	2.50%	6.99%	5.36%	8.33%	4.57%	3.78%
Beer – Recall	6.82%	20%	6.82%	17.5%	32.95%	7.95%
Beer – F1 Score	3.68%	10.35%	6%	11.29%	8.03%	5.13%
Negative – Precision	61.11%	59.80%	72.87%	68.55%	40.77%	67.67%
Negative – Recall	74.88%	72.81%	74.16%	73.44%	73.86%	74.80%
Negative – F1 Score	67.30%	65.67%	73.50%	70.91%	52.54%	71.06%
Other Phones – Prec.	9.52%	14%	2.72%	4.67%	10.20%	6.80%
Other Phones – Recall	12.07%	9.01%	6.56%	5.79%	8.88%	10.10%
Other Phones – F1 Sc	10.65%	10.97%	3.85%	5.17%	9.49%	8.13%
Samsung – Precision	24.19%	8.24%	17.20%	13.19%	9.14%	19.35%
Samsung – Recall	15.10%	10.27%	11.85%	12.24%	9.14%	13.95%
Samsung – F1 Score	18.60%	9.15%	14.04%	12.70%	9.14%	16.21%

Table 1: Precision, Recall and F-1 Scores summary of iterated models

3.3 Model Evaluations and Results Analysis

Data: I have learned by labeling the data, that there was a very unique video in the test set very different from the typical ads in the training data. Thus, after checking with SB, this video file was considered an out-liar, being cleansed out of the 'Test Set'. In addition, to tackle data imbalance I weighted each class by the inverse of respective percentages in the training set. I also used augmentation techniques: zooming (up to 50%), horizontal flipping and rotation (up to 40 degrees).

Model: Given time constraints to this project delivery is wasn't possible to near human performance error levels of 2-5%. It is necessary to perform more training iterations with deeper models. However, the data provided to this project seems to have an accuracy ceiling of around 90% percent applying similar architectures to the ones used in this piece.

When I interpreted the results among the four classes shown in Table 1, f1-scores on the classes that really matter to this problem solution are not satisfactory. Such results on the different classes predictions highlight that without significantly more data there is little room for improvement applying an end-to-end deep learning approach to single frames classification.

4 Conclusions and Future Work

In order to gain significant improvements in performance to reach human level, hand engineering in the data set and problem solving approach seems to be a good alternative going further. Because of time constraints at this point, I adopted an end-to-end approach, since I had to undertake efforts preprocessing and labeling the entire data from video files to label image frames. Thus, to solve this problem using end-to-end deep learning it is necessary to train the developed learning algorithms with significantly more data. Finally, it is worth to state that results achieved in the literature were not far from the ones reported in this piece, hence they also help to substantiate hand engineering into smaller tasks, breaking the bigger problem into less complex activities performed by specific algorithms.[2][4][6]

Despite not yet nearing human performance to automate SB related processes, the objectives of this project were partially met since I consider this research on real world data and problem as a stepping stone diagnosis on the problem and available data. Moving forward, I recommend prioritizing among the following alternatives:

1. Pursue image frames annotations on the products being advertised in each image similarly to an object detection project with sliding windows detection and bounding boxes anchoring implementing YOLO algorithm;
2. Significantly gather more data to include in this presented end-to-end deep learning approach. However, video storage is very expensive in the current state of technology, thus it should be considered among the criteria when prioritizing;
3. Implement architectures implemented on literature to train a deeper networks that consider spatial time fusion to train the data. This approach would require more data available than the rough 500 video files used in this project. For instance Andrej Karpathy's paper on the Youtube sports videos trained on 18,000 sports videos, which were also longer than the average 30s advertises used in this project.

References

[1] https://shoppingbrasil.com.br

[2] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar & Li Fei-Fei (2014) *Large-scale Video Classification with Convolutional Neural Networks.* http://cs.stanford.edu/people/karpathy/deepvideo

[3] Doërr, Gwenaël, and J-L. Dugelay. "Security pitfalls of frame-by-frame approaches to video watermarking." (2004) *IEEE Transactions on Signal Processing 52.10 : 2955–2964* & https://www.ffmpeg.org

[4] https://medium.com/@alibugra/extract-frames-as-an-image-from-video-files-using-ffmpeg-65b52d3d97db

[5] http://www.shoppingbrasil.com.br/portal/modules/digitalizacao/index.php?idseq=06974972&idprdt=1961355

[6] A. Krizhevsky, I. Sutskever, and G. Hinton (In NIPS, 2012) *Imagenet classification with deep convolutional neural networks.*.

[7] https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/

[8] https://www.tensorflow.org/tutorials/images/classification

[9] https://www.tensorflow.org/api_docs/python/tf/keras/applications/VGG16

[10] https://www.tensorflow.org/tutorials/images/classification