Screenshot Classification Using Convolututional Neural Networks

Raja Asim ASIMRAJA123ASIM@GMAIL.COM
Department of Computer Science
FAST-NUCES Lahore
Lahore, Pakistan

Editor: None

Abstract

In this paper I present that Convolutional Neural Networks can be used to classify screenshots among different categories. It can be used in any application whose task is to take screenshots and then later a model can add them in their respective categories or folders and hence the performance or time graph of an employee can be made. The dataset oftenly used in deep learning contains lots of images and it sometimes takes hours or even days for training the model. I used 1420 images from Ubuntu for 6 categories i.e. 236.67 screenshots on average for each of the category and the model was trained in almost 15 to 20 minutes when the epochs were 20. The training accuracy achieved was 97.26% and the validation accuracy was 94.5%. Of course the model did misclassify some of the test screenshots taken but when they were given their correct labels and shuffled with training data then the model was trained again tested on the same data on which it was tested before it didn't misclassify the same image again. The number of classes were 6 and the model was tested on Windows' screenshots and had accuracy of 83.6% there. But some classifications are just too much similar that even humans can't judge their class and my model also did the same for youtube's and facebook's fullscreen videos. Every time a full screen video from either of the two categories was tested, the model classified it as one of these two classes, and not in any other.

Keywords: Screenshot Classification, CNNs, Approaching human level performance

1. Related Work

CNNs have revolutionized the image classification task and some research and advancement has also happened in the field of CNN for screenshot classification (Anand Sampat, 2015) for employee monitoring or supervising. Often in deep learning (LeCun et al., 2015) models the dataset used is quite large but my method has a very small dataset i.e. 1420 images. Also the accuracy achieved is quite high and the model trained on Ubuntu's screenshots can be tested on Windows screenshots and it does a great job even there. So my model will generalize even well if it has seen Windows' screenshots in the training but it did OK because there isn't a lot of difference between Windows' and Ubuntu's screenshots.

CNNs are being excessively used in the recent researches where people are beating the previous sate-of-the-art results like in scene classification (Zhou et al., 2014). With CNNs we don't have to worry about the low level features, it automatically figures it out. Also

Google got a relative improvement of 70% on unsupervised learning for image classification (Le, 2013) on the previous state-of-the-art. Almost all the methods used in classification had lots of data and intensive computations. CNNs are also being used in scene classification, my model is a specialized category of scene classification where the scene is actually my laptop's current screen.

2. Introduction

CNNs (LeCun et al., 1999) are remarkably good at image classification, but have we tested them on the screenshots of facebook, gmail, stack overflow, coding editors/coding softwares and all other categories i.e. browsing internet, using local file system etc? Screenshot classification is a lot like scene classification but the scene is of various types. Same facebook will have a slightly different layout or picture on Ubuntu and will have different theme in Windows or Mac.

The primary motivation for this work was that we can automate the classification of screenshots taken by any application so that the manager or the employer can have a good view of each employee's efforts and can see that is the organization going on the right path or not and every employee is performing or doing his/her work as expected or not. Also since not a lot of work has been done in classifying screenshots so I thought to check out a really simple CNN model on identifying screenshots.

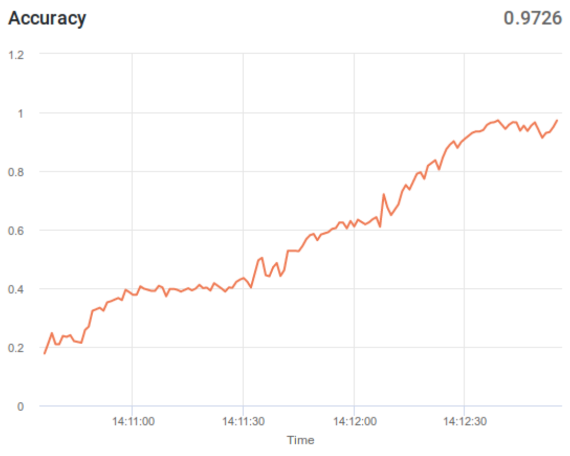

Figure 1: The training accuracy with 20 epochs

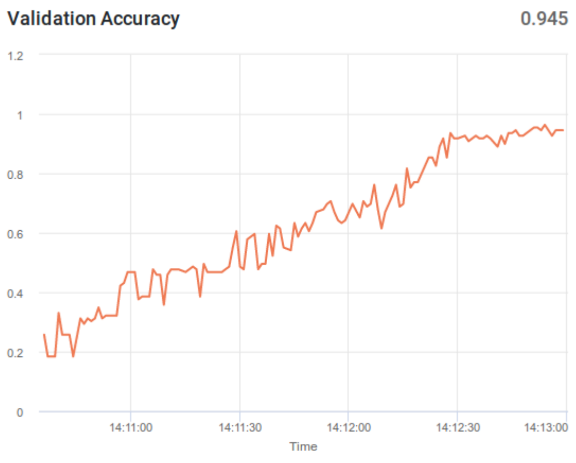

Figure 2: The validation accuracy with 20 epochs

3. Training and Testing Data

The training data consisted of screenshots taken randomly by a node js application that just takes screenshot after some interval which in my case was 2-5 seconds. Screenshots were taken of facebook, youtube, gmail, stack overflow, code editors, coding softwares and others. Since in all cases except code editors the data variation was very small so it was easy for Neural Network (Haykin and Network, 2004) to generalize on other examples and it did well with less examples. After assigning labels to each training example the model was trained, it was tested on Ubuntu's and Windows' screenshots. There were 236 on average screenshots for each category.

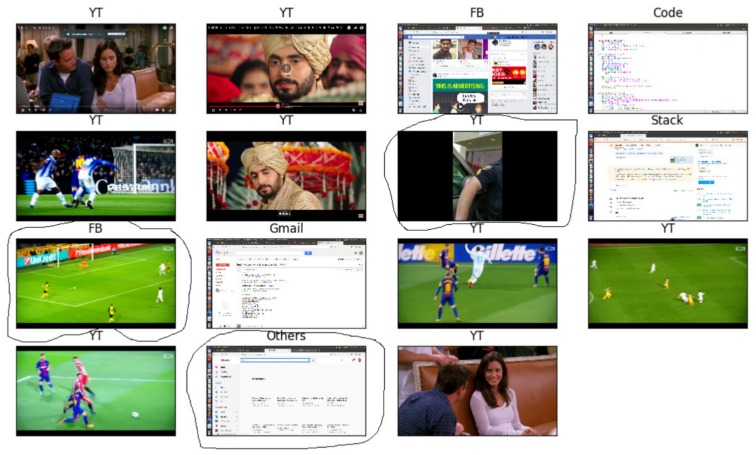

Figure 3: Ubuntu screenshots being classified, the circled figures are wrongly classified.

The testing data was taken by the same node js application that randomly took screenshots and added it in the test folder. All the training set data comprised of just Ubuntu's screenshots. The testing data later did have Windows' screenshots as well to check the relative improvement of the model.

4. Methodology

I used a 5 layered CNN with 32,64,32 filters in the convolututional layer and each having the filter used of size 5x5, padding was same, activation function was ReLU and no stride was used. Max pooling layers after each convolututional layer was also used and the filter used there was also of 5x5. Then I had a fully connected layer of 128 units and the activation function was also ReLU (Dahl et al., 2013) here. Finally the output layer had the softmax activation function and since the classes to classify the screenshot were 6 so the hidden units in the output layer were also 6. The dropout (Srivastava et al., 2014) was also used and the keep probability for each neuron or the hidden unit was 0.4. The optimizer used to minimize loss was Adam optimizer (Kingma and Ba, 2014) with the learning rate of 0.005, the loss was categorical cross entropy (Hoskisson et al., 1993). One hot encoding scheme was used for labels. The image was multi color, resized to 50x50 image. So the dimensions of the image were 50x50x3.

Figure 4: Facebook screenshot being classified through different epochs.

The training and validation accuracy can be seen while training the model and testing accuracy was calculated by seeing the misclassified images present in the folder after the testing phase has been done. On increasing epochs upto 20 the accuracy increased correspondingly but after 20 epochs the validation and training accuracy tend to increas very slowly so I used early stopping (Prechelt, 1998) to stop at 20 epochs. To prevent overfitting dropout was used and also epochs weren't increased further because of the fear of overfitting (Hawkins, 2004). The cross validation size was 0.1 of the training data size i.e. 142 sample pictures. I trained and tested my model on the online servers on Floydhub [1] on the powerful GPU version so the whole model was just trained in 15-20 minutes and the accuracies were shown.

1. floydhub(http://www.floydhub.com/)

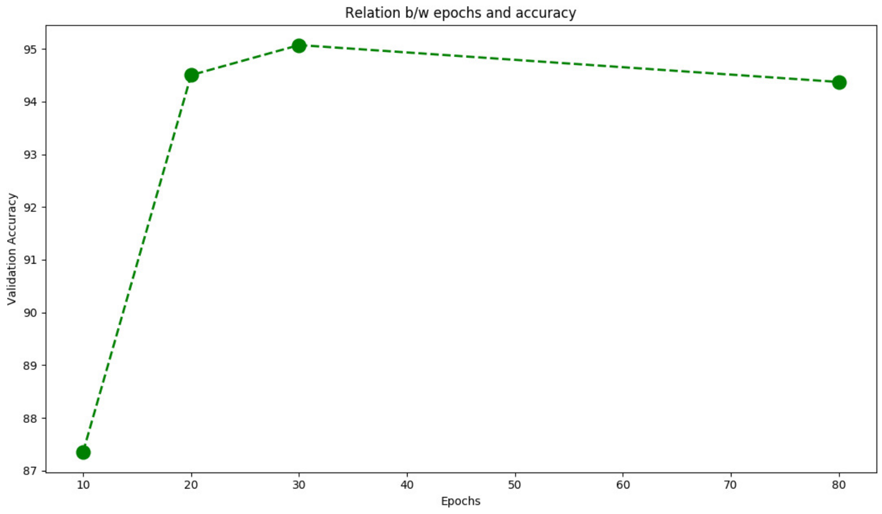

Figure 5: Validation accuracy on different epochs

5. Drawbacks

The neural network had no way to know whether the video screenshot is from facebook or youtube because each of them have fullscreen video images in the training data. But if hints are given like the video progress bar shown and the buttons below were shown then the network predicted the correct class but otherwise it was really just a coin toss between adding in facebook's folder or in youtube's.

I think that there is no way to train a network in such a good form that it can classify perfectly the videos of facebook and youtube when they are in full screen mode. Because the categories of videos in both the classes are almost same and the network just sees a full screen picture which can be part of facebook or maybe youtube but of none other.

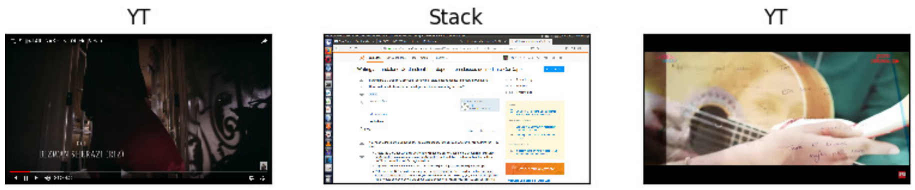

Figure 6: Youtube screenshot being classified by looking at video progress bar and other hints.

6. Learning over Misclassification

All the training data with labels comes into a folder and then the model learns from labels and is trained using CNN. For testing, the data came into one folder and all the images were unlabelled, after predicting class for each screenshot the image is added in the corresponding folder i.e. facebook folder will have all screeshots of facebook and youtube folder will have all screenshots of youtube present in the test data etc when the testing phase is done. But since the accuracy isnt 100% some of the images were misclassified and were assigned wrong classes and hence folders. So all the misclassified images were sent to their correct folders by some hand work and after spending some time. Then all the images that were 100% correctly labelled were mixed up with my training set present initially and the model was trained again. Now the model had higher accuracy because it had learnt not to misclassify images like previous ones again so by learning over misclassification the accuracy of my model increased gradually. The testing accuracy varies from 89-92% for my naive classifier but if learning over misclassification is applied it increases but very slowly. Maybe it'll work better if the dataset is quite large and then the model will learn from it's mistakes quite easily.

Table 1: Results of Classification

Platform	Training Size	Testing Size	Validation Accuracy	Testing Accuracy
Ubuntu	1420 images	270 random images	94.5%	92.23%
Windows	-	140 random images	-	83.6%

7. Testing on Windows' Screenshots

Since all the training data had Ubuntu 16.04's screenshots so I thought maybe it won't generalize good enough on Windows' screenshots but it turns out that I was wrong there. Since the dataset of Ubuntu was quite similar to Windows so I got the accuracy of 83.6% when the screenshots of Windows of same categories were tested on the very same model that was just trained on Ubuntu's screenshots. Since the model hadn't seen and learnt from Windows' screenshots that it did see in the training so the accuracy relative to Ubuntu's testing was low.

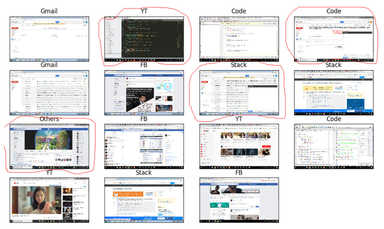

Figure 7: Classification of Windows' screenshots. The circled figures are wrongly classified.

8. Conclusion

With a small dataset like 1420 images I got the training accuracy of 97.26% and the training loss of 0.26, the validation accuracy of 94.5% and the validation loss of 0.30 on 20 epochs. This means that the CNN is good enough to take hints from the screenshots and identify them correctly. I believe that with enough data it can even match the human level accuracy atleast in identifying the screenshots. But in some cases even we humans can't correctly tell that whether this image is from facebook or from youtube so the CNN did commit mistake in classification of such images also. I did classification on 6 categories but with data enough fed any number of classifications can be made and with some hardwork and labeling correctly the misclassified images and then training the model again does reduce some classification errors. In the table shown below the correctly and wrongly classified screenshots of each category are shown.

Table 2: Testing Results on Ubuntu

Category	Correct	Wrong
Facebook	63	2
Youtube	64	1
Stack Overflow	16	4
Gmail	39	5
Coding	57	6
Others	25	7

In the above table, the testing results on Ubuntu are shown. To see if the screenshot was added in the right category or not some handwork was done. Total images were 289 and the misclassified images were 25. So the testing accuracy is 91.35%.

CNNs can make the employee monitoring job easy. They can classify the screenshots with a great accuracy when fed with even little data i.e. 237 images on average for each class. Since the variation isn't that much large in all the screenshots of one class so with even small dataset one can use CNN to supervise the employee's performance. And even though if the model commits some silly errors and misclassifies some of the screenshots one can label them correctly and train the model again by adding them in the training data so that the accuracy of the model can be improved. Also the model used to classify the screenshots was very small as compared to other deep networks used to classify the images. The network used by me just had 3 convolutional layers, 3 max pooling layers, 1 fully connected and 1 output layer. The total hidden units used were 256 and 6 output units. So, a small CNN can be used to classify screenshots in some categories with a great accuracy.

References

Avery Haskell Anand Sampat. Cnn for task classification using computer screenshots for integration into dynamic calendar/task management systems. 2015.

George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE, 2013.

Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

Simon Haykin and Neural Network. A comprehensive foundation. *Neural networks*, 2(2004): 41, 2004.

Robert E Hoskisson, Michael A Hitt, Richard A Johnson, and Douglas D Moesel. Construct validity of an objective (entropy) categorical measure of diversification strategy. *Strategic management journal*, 14(3):215–235, 1993.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Quoc V Le. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE, 2013.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.