

Deep Triage: Smart photo album sifter using Transfer learning and Siamese networks

Digvijay Karamchandani
Department of CSE, UCSD
PID: A53220055
dkaramch@eng.ucsd.edu

Kriti Aggarwal
Department of CSE, UCSD
PID: A53214465
kriti@eng.ucsd.edu

Karan Uppal
Department of CSE, UCSD
PID: A53217553
k1uppal@eng.ucsd.edu

ABSTRACT

Messengers today have a plethora of junk and redundant images. The search of a relevant image is almost like finding a needle in a haystack. A messenger album essentially comprises of two types of images, documental and natural. Document images can range from being junk (memes/jokes/greeting) to extremely important(scanned documents). On the other hand, natural images are often taken in a series of nearly redundant pictures to capture a moment or scene. However, selecting photos to keep or share from a large collection is a painful chore. To address these problems, we take a two-step approach at first we separate out the document and natural images. We further proceed with this split dataset to cluster the unimportant document images by category and seek a relative quality measure within a series of natural images taken of the same scene, which can be used for automatic photo triage. For the clustering task we achieve an accuracy of 91.41% and 89.52% on the train and test set respectively. For the more challenging automatic photo triage task we achieve an accuracy of 70% on the test set.

1 INTRODUCTION

The idea of separating and clustering out junk images and extracting the most relevant image from a burst of redundant images can be framed as a classification and triaging problem respectively. For the first task, we followed a hierarchical classification approach. We trained a CNN which learnt to classify between document and natural images, and then to further cluster out the document images ranging from unimportant to important classes. We fixed on four classes (scanned documents, jokes and memes, greetings and natural images).

The second problem of automatic photo triaging in natural images was a much more challenging task. For this we used the concept of Siamese networks, with each individual network being a pretrained VGGNet smoothened to natural images. The Siamese network takes in two images as inputs and outputted the probability pair having the probability of each image being better. The output is then passed through a 2 layer Perceptron to output 1/-1 corresponding to the first or second image being better respectively.

2 LITERATURE REVIEW

The Smart Photo album Junk Photo Classification task was novel in it's own way as there was no specific literature to follow for this task. Evidence that CNNs could prove useful for the problem came from our own experiments with varied layered CNNs. This lead us to explore the performance of state of the art CNN architectures which gets pushed up slightly every year with the improvements in the complexity and computation of the architecture. However, the

main drawback of latest state of the art is ever increasing traintime to get the models to convergence. To both take the advantage of the features learnt by these models and saving time on training. The model that has been widely used in the past years for application problems has been the VGGNet architecture, which won the ImageNet Challenge in 2014. It benefits from having less layers than the current state-of-the-art while still having significantly good with an error of 7.3% on accuracy [11]. VGGNet models are also widely distributed on the Internet for various deep learning frameworks, making it an ideal CNN model to start out with[12].

Within the field of Convolutional Neural Networks, there has been a lot of work done with Transfer Learning. Transfer learning is useful when one wants to train a CNN on their own dataset, but for various reasons, the dataset may not be adequate to train a full neural net on (i.e. it could be too small). While data augmentation is a viable option in a lot of cases, transfer learning has also proven effective. Transfer learning refers to the process of taking a pre-trained CNN, replacing the fully-connected layers (and potentially the last convolutional layer), and training those layers on the pertinent dataset. By freezing the weights of the convolutional layers, the deep CNN can still extract general image features such as edges, while the fully connected layers can take this information and use it to classify the data in a way that is pertinent to the problem. Since, the dataset for the classification and clustering of junk images was small, transfer learning suited our purpose. Hence, we chose to do transfer learning using pre trained VGG Net with additional layers of convolution for performing our first task.

For the second part of the project that involved selecting the best picture from a series of nearly redundant pictures to capture a moment or scene, we followed the architecture described by Huiwen Chang et al. in the paper 'Automatic Triage for a Photo Series'[1]. We also reviewed other related works on image summarization by Sinha et al.[14] and Simon et al.[13] that jointly captures photo quality, event coverage and scene diversity but found that the task of photo triaging differs from the image summarization problem. Since, the triage problem we address can be viewed as a sub-problem of image summarization, as it focuses only on finding good images from photo series, but does not consider the representativeness of these images to the whole album[1].

3 DATASET

For the task of classifying and clustering junk images from all image categories images, we had no any reference dataset with us. Also, the choice of junk or useless images is highly subjective and depends on the regular users choice. Hence, we performed a survey to find the usual categories of images in which users would likely

Table 1: Experiments with Different Number of Classes

Classes	Validation Accuracy	Training Accuracy	Architecture
2(scan,greet)	95.2	96.5	1 conv (80 units)layer
3(scan,meme,greet)	88.1	89.5	1 conv layer (80 units)
4(scan,meme,greet,miscellaneous)	52.25	52.5	3 Layered Conv-Nets

Table 2: Experiments with different Number of Hidden Layers

Architecture	Validation Accuracy	Training Accuracy
VGGNet + 2 Hidden Layers(1024 + 512)	91.21	93.5
VGGNet + 2 Hidden Layers(100 + 50)	88.4	89.5
VGGNet + 2 Hidden Layers(200 + 100)	88.82	89.98
VGGNet + 2 Hidden Layers(600 + 300)	89.54	90.86
VGGNet + 2 Hidden Layers(1000 + 500)	91.23	92.1
VGGNet + 2 Hidden Layers(1400 + 700)	90.34	91.2

find their most of their Junk images.

We created a Google form and asked them to list down the top 5 categories of images in their phones or any other storage devices and also mark the categories which they would likely delete if they had a storage crunch. Around 90% of the users came up with these 4 broad categories:

- (1) Memes
- (2) Greetings
- (3) Scanned Documents
- (4) Miscellaneous (like Duplicate Images, Burst images,etc)

Out of which most users marked memes and greetings as the junk images. Users were divided on deleting miscellaneous and scanned documents images.

To build our dataset, we built a **Web Scraper** to scrape around 10,000 images from websites like:

- (1) **Memes:** <http://knowyourmeme.com/>
- (2) **Memes:** <http://imgur.com/>
- (3) **Greetings:** Google search results for different greeting images.
- (4) **Scanned documents:** <http://www.umiacs.umd.edu/~zhugy/tobacco800.html>
- (5) **Scanned Documents:** <http://tc11.cvc.uab.es/datasets/GenderIdentify2013>
- (6) **All categories:** Many photo samples from Friend's Phone Images .
- (7) **Miscellaneous:** Adobe Triage Dataset - <http://phototriage.cs.princeton.edu/dataset.html>

For Photo Triaging we used Adobe Triage dataset and used around 15,545 unedited photos organized in 5,953 series based on timestamps.

3.1 Pre-processing

Since, the dataset was created by scrapping images on the internet, there was lots of noise in our dataset. To reduce the noise we had to manually prune out images especially from the memes and greetings categories. For the scanned documents, we were able to

combine various different datasets like handwritten texts, scanned documents, scanned newspapers etc.

Since, we used the VGG Net architecture, we also normalized the images as is performed in the VGGNet paper. This simply involved applying normalization by subtracting the mean of all images from each image. Additionally, since photos came in varying sizes and resolutions, we used OpenCV to scale all the images to one size. We initially started with an ambitious image size of 256 X 256 X 3 for training, however that did not work out well for our case. Since, it took a lot of time to train on such big images. To combat that with limited time constraint, we planned to reduce the image size to 32 X 32 X 3, which performed well in our case and gave convincing results.

4 ARCHITECTURE AND EXPERIMENTS

4.1 Architecture and Experiments for Junk Image Classification

4.1.1 Experiments with different Number of Classes. We initially started with 1 layer Conv-Nets with 80 activation units and were successfully able to classify two classes below with a Validation Accuracy of 95.2% that is -

- (1) Scanned Documents
- (2) Memes and Greetings

This can be because the categories scanned documents and memes & greetings are very different and just one convolutional layer is sufficient to segregate the 2 classes.

However, when we tried the same network on 3 classes by so that the model can differentiate between memes and greetings that is -

- (1) Memes
- (2) Greetings
- (3) Scanned Documents

We got a lower Validation accuracy of around 88.1%. So, we increased the number of layers in our model from 1 to 3 layers and tried to run our model on the 4 classes below-

- (1) Memes

Table 3: Training and test accuracy with various architectures

Classes	Validation Accuracy	Training Accuracy	Architecture	Iterations
3(scan,meme,greet)	88		1 conv layer (80)	10
2(scan,greet)	95		1 conv (80)layer	10
3(scan,meme,greet)	86.64	85.54	2 conv (150, 80)layers	30
4(flicker photos)	52.25	52	3 layers()	10
4(vggnet) Flicker dataset	88.19	97.12	2 extra dense (1024,512)layers	16
4(vggnet) Adobe dataset	88.14	97.34	2 extra dense layers	16
4(vgg net) + dropout (0.5)	89.52	91.41	2 extra dense layers	16

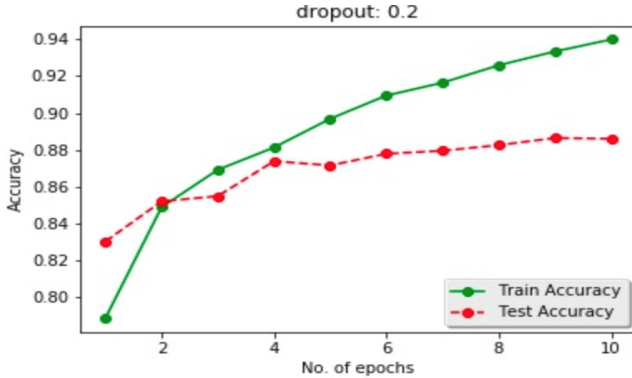
Table 4: Experiments with different Dropouts

Dropout	Validation Accuracy	Training Accuracy
0.2	89.8	93.2
0.4	90.9	92.1
0.5	91.21	93.5
0.6	90.8	92.3

- (2) Greetings
- (3) Scanned Documents
- (4) Miscellaneous Pictures

However, after adding the new class of images that is Miscellaneous Pictures which were natural images of humans, objects and animals, our model performed poorly with a validation accuracy of only 52%. This can be attributed to the fact that the miscellaneous pictures is a complex class and its hard to learn its features with our simple architecture and only 8000 training samples. To deal with issue, we moved to transfer learning which combines the advantages of learning complex features using pre trained complex CNN architectures in very less training time.

We used VGG16 Network since it has lesser layers and still provides an accuracy of approximately 7.3%. We added two additional fully connected layers to learn the features of other classes part of our categorization problem. To finalize on the number of layers,hidden units etc. to be used in the layers after the VGG Net, we ran a number of experiments presented below.

**Figure 1: Dropout experiment with a Dropout=0.2**

4.1.2 Experiments with Number of Hidden Layers and hidden units. To choose the number of hidden layers in the CNN, we planned a basic approach where we ran our model with different number of neurons in the hidden layer and cross validated the model with different configurations and got the average MSE, then by plotting the average MSE vs the number of hidden neurons we saw that which configurations are more effective at predicting the values of the test set and then finally dig deeper into those configurations only, therefore possibly saving time too. Hence then we picked the optimal number of nodes in each hidden layer keeping in mind the following -

- (1) Too few nodes will lead to high error as the predictive factors might be too complex for a small number of nodes to capture.
- (2) Too many nodes will over fit our training data and not generalize well.

Hence we ended up choosing the value of 1000 neurons in first hidden layer and 500 neurons in the second hidden layer.

4.1.3 Experiments with Dropout. Dropout is a regularization technique for neural network models, where randomly selected neurons are ignored during training. We started with a small dropout value of 20% and slowly increased the dropout rate to around 60% to provide regularization to our model. We saw that we got the optimal value at around 50% where the Training accuracy and Validation accuracy were both optimal.

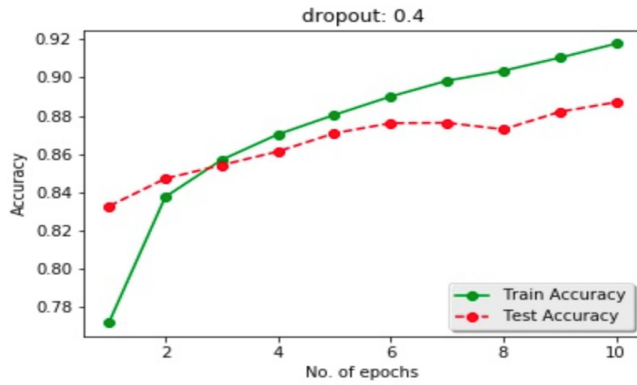


Figure 2: Dropout experiment with a Dropout=0.4

4.2 Final Architecture for broad Image Classification

The final architecture for the broadly classifying the images in the 4 categories

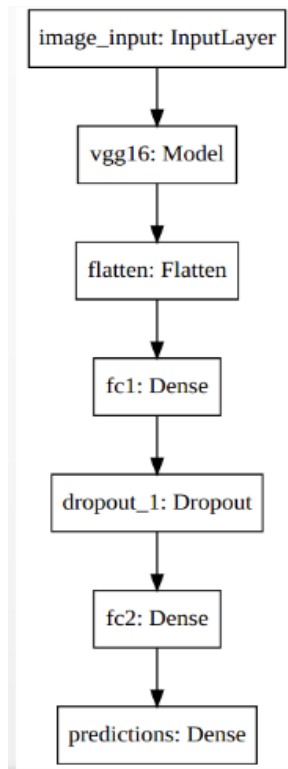


Figure 3: Final architecture for image classification

Below are the confusion matrix and some of the mislabelled examples obtained using this network.

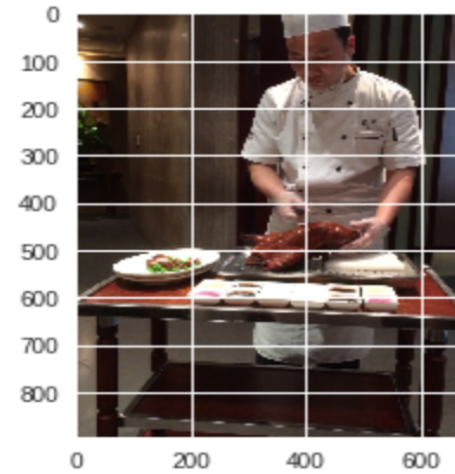


Figure 4: Predicted Label: Greetings, Correct Label: Miscellaneous

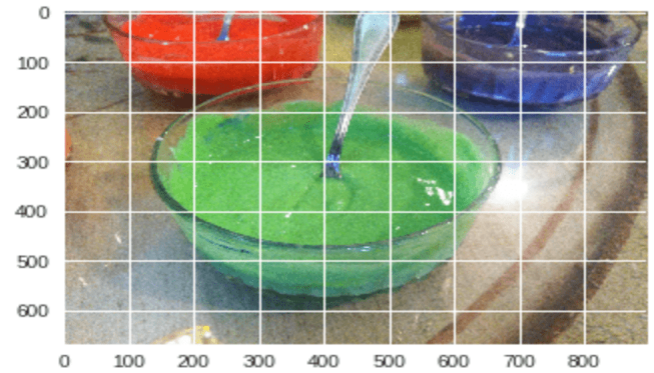


Figure 5: Predicted Label: Greetings, Correct Label: Miscellaneous

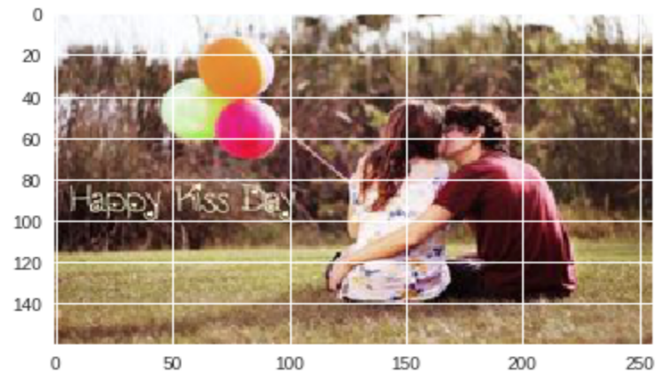


Figure 6: Predicted Label: Miscellaneous, Correct Label: Greetings

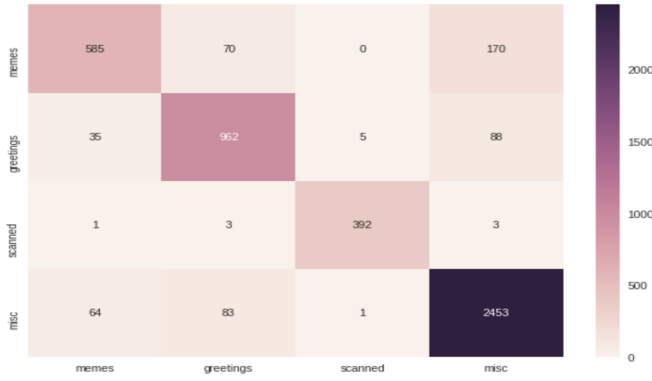


Figure 7: Predicted Label:Memes, Correct Label:Miscellaneous

4.3 Architecture and Experiments for Photo Triaging

The Siamese network takes in two images as inputs outputs the probability pair having the probability of each image being better than the other image. The paper[1] makes use of the following pipeline for Photo Triaging:

- (1) Sample the image and pass the same image through two Siamese networks in parallel which are configured in a similar fashion.
- (2) Each of the Siamese Networks is a Conv-Net initialized with a VGG-net weights.
- (3) We can calculate the difference in the image output results from both of the networks.
- (4) Pass the result through a 2-layer multi-Perceptron

The Siamese network pipeline is shown in Figure8

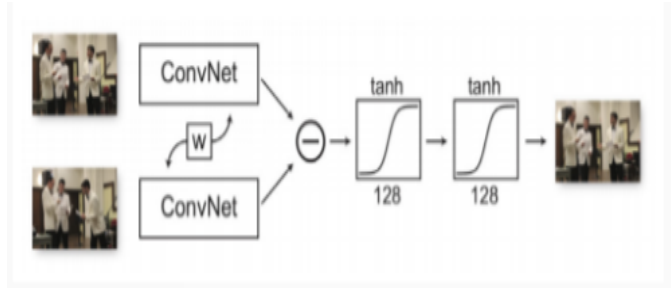


Figure 8: Siamese network architecture

Since, no code was available for this method, we modified our existing architecture that we used in the initial broad image classification in to a Siamese network. Though, due to lack of time and resources we could only experiment with training the last layers of the VGGNet. Hence, we failed to achieve good results using this architecture.

Hence we moved onto use the pre trained Caffe Model provided with the paper and ran our further experiments on this model.

4.3.1 Experiments with different Series size. We experimented with the length of series or number of photos in a burst. A burst of photos is defined using the image metadata such as timestamp. Photos with in a threshold of timestamp is called as a burst as described in the paper. We experimented with how the probabilities of the same image pair changed with the change in the number images in a series. The results were quite surprising as the probabilities changed for the same image pair while being a part of the series of varied length. Though the probabilities changed, the change was not so big so as to change the winner in a series of images.

4.3.2 Experiments with different input order of images. The paper claimed that the results of the network was skew symmetric i.e. if an image pair returns the output 1,-1 that is left image is the winner. Flipping the two images should return -1,1. We decided to test this claim by selecting two images in a burst and passing them through the network. We then flipped the images and again passed them through the network. We found that the outputted probabilities changed with the flip. Following are the results of this experiment.

We found that though the probabilities obtained were not exactly the same, the difference in the probabilities was minor and did not impact the winner in an image pair.

4.3.3 Custom Heuristic. The paper used pairwise comparison results to obtain a global ranking for each image in a series. The pairwise comparison annotations are denoted as a count matrix $S = s_{i,j}$, where $s_{i,j}$ is the number of times that the i th photo I_i is preferred over the j th photo I_j . Bradley-Terry model[15] is used for obtaining the global ranking, which describes the probability of choosing I_i over I_j as a sigmoid function of the score difference between two photos $i, j = c_i - c_j$, i.e.

$$\begin{aligned}
 P(I_i > I_j) &= F(\Delta_{i,j}) \\
 &= (e^{\Delta_{i,j}}) / (1 + e^{\Delta_{i,j}})
 \end{aligned}$$

The pairwise winner is calculated using the Bradley-Terry model described above.

The series level evaluation measures the log likelihood orderings over all series as follows. For a series k , given the human preferences on each image, we identify the one most preferred by humans as $winner(k)$. For a method M , we apply it to all image pairs that include $winner(k)$ and another image in that series. Next we compute the logarithm of the joint probability of the decisions of method M on all such pairs.


$$logL_k(M) = \sum \log P(M, winner(k), i),$$

where $P(M, i, j) = Pr(I_i > I_j)$, M predicts photo i better than j $Pr(I_j > I_i)$, otherwise

The paper was choosing the winner image from a burst of images in a series using the method described above. This method completely disregards the number of times a image wins from other images in a series. We came up with a heuristic to combine these two ideas by representing the comparisons between images as matches in a tournament as follows:

- Select the image chosen by the previous model as winner. Call this image W .

Table 5: Order of images and corresponding probabilities of that image being better.

Order of images	Probabilities of the corresponding image being better
	<p>(0.49800199, 0.50199795)</p> <p>(0.50375259, 0.49624738)</p>

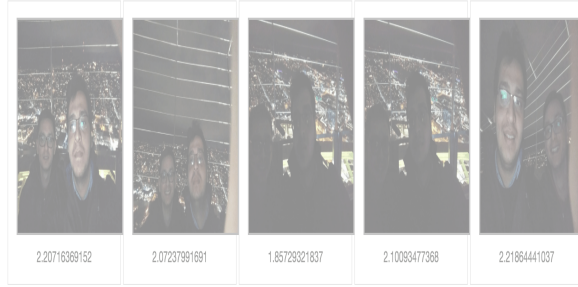


Figure 9: Burst3 without Heuristic with Probability label

- Find all the images who won from W in a pairwise comparison using Bradley Terry model. Call these images as competitors.
- Calculate the matches each image won in pairwise combat. If the matches won by any of the competitors is more than W , make it the winner.
- If the previous step results in more than one winner, break tie using Bradley-Terry model.

We were able to achieve good results in some of the image series. One such image burst is shown below in Figure 9 with the calculated log probabilities below each image. The model given in paper selects the first image while our heuristic makes the last image as winner. The differences in the image is subjective, though can be easily seen in this case.

5 CONCLUSIONS

In this project, we essentially performed a classification and a selection task. We were able to solve the problem of creating a smart album viewer right from collecting dataset to segregating the images in categories to selecting the best images in a burst of images in the album.

We studied and explored different training mechanisms such as transfer learning, pre-training and optimizations for the first task of clustering different types of images. For the second task, we studied and explored how the old network could be molded

into a Siamese network for comparing and finding the most relevant features in the pair of images, and decided on the better image.

For classification, we experimented with various CNN models. While vanilla CNN with even 5 layers did not work well with the data Accuracy: 52%. On the other hand, transfer learning by tuning VGG net with two fully connected layers on our dataset worked well for the first task, and we achieved an accuracy of approx 91.3%.

For auto-triage, multiple experiments were performed in training the number of layers in VGGNet, series size, order of input images and winner heuristics. We used the pretrained model provided in the paper and applied various heuristics on top of it. We could reproduce slightly better results than the paper in certain cases, but could not improve upon the accuracy presented.

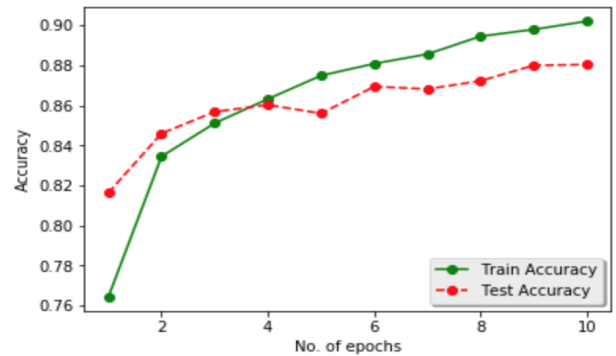


Figure 10: Training and test accuracy with 2 layers(512,1024)

6 REFERENCES

- [1] <http://phototriage.cs.princeton.edu/paper/Chang2016APT.pdf>
- [2] <https://en.wikipedia.org/wiki/Airbnb>
- [3] https://en.wikipedia.org/wiki/Decision_tree_learning
- [4] <http://insideairbnb.com/get-the-data.html>
- [5] <https://www.kaggle.com/residentmario/d/airbnb/boston/modeling-prices/>

Layer (type)	Output Shape	Param #
input_3 (Input Layer)	(None, None, None, 3)	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1792
block1_conv2 (Conv2D)	(None, None, None, 64)	36928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73856
block2_conv2 (Conv2D)	(None, None, None, 128)	147584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295168
block3_conv2 (Conv2D)	(None, None, None, 256)	590080
block3_conv3 (Conv2D)	(None, None, None, 256)	590080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1180160
block4_conv2 (Conv2D)	(None, None, None, 512)	2359808
block4_conv3 (Conv2D)	(None, None, None, 512)	2359808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2359808
block5_conv2 (Conv2D)	(None, None, None, 512)	2359808
block5_conv3 (Conv2D)	(None, None, None, 512)	2359808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0

Figure 11: Final Model

Layer (type)	Output Shape	PARAM #
image input (Input Layer)	(None, 32, 32, 3)	0
vgg16 (Model)	multiple	14714688
flatten (Flatten)	(None, 512)	0
fc1 (Dense)	(None, 1024)	525312
dropout_1 (Dropout)	(None, 1024)	0
fc2 (Dense)	(None, 512)	524800
predictions (Dense)	(None, 4)	2052

Figure 12: Caffe Model

[6] Winkler, R.L. and Makridakis, S. (1983). The Combination of Forecasts. J. R. Statis. Soc. A. 146(2), 150-157.

[7] Makridakis, S. and Winkler, R.L. (1983). Averages of Forecasts: Some Empirical Results. Management Science, 29(9) 987-996.

[8] <https://www.cs.cornell.edu/home/cardie/papers/masa-sentire-2011.pdf>

[9] <http://res.cloudinary.com/general-assembly-profiles/image/upload/v1474479496/geovpl78jbsx7dzqycde.pdf>

[10] https://en.wikipedia.org/wiki/Support_vector_machine

[11] K. Simonyan and A. Zisserman (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR.



Figure 13: Burst1 without Heuristic with Probability label

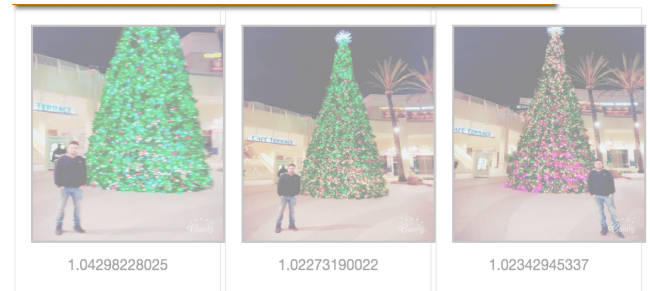


Figure 14: Burst2 without Heuristic with Probability label

[12] Diveesh Singh and Pedro Garzon. Using Convolutional Neural Networks and Transfer Learning to Perform Yelp Restaurant Photo Classification.

[13] SIMONYAN, K., AND ZISSERMAN, A. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[14] SINHA, P., MEHROTRA, S., AND JAIN, R. 2011. Summarization of personal photologs using multidimensional content and context. In Proceedings of the 1st ACM International Conference on Multimedia Retrieval, ACM, 4.

[15] RALPH ALLAN BRADLEY, M. E. T. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika 39, 3/4, 324-345.

6.1 Model and Demo Code Links

- (1) Model Code - <https://github.com/kritiagg/Smart-album-viewer>
- (2) Demo Code - <https://github.com/karanuppal2790/Smartalbumviewer>
- (3) Presentation Link - <https://prezi.com/p/sfns8i.m3li5/>