CS230

Is a Picture Worth 1,000 Words? Visual Emotional Analysis using Transfer Learning and CNNs

Katherine Kowalski kasia4@stanford.edu
Noah Jacobson noahj08@stanford.edu
Hasna Rtabi hasna@stanford.edu

Abstract

There is strong psychological evidence that images tend to elicit certain emotions in their viewer based on the style and the content. As social media networks become more widespread, there are more use cases for neural networks that analyze emotions based on image data. Our team tried three approaches to create a neural network that could label an image with one of eight emotions: amusement, anger, awe, fear, sadness, happiness, contentment, or excitement. Our first model was a fractal neural network, which was unsuccessful because it was extremely computationally expensive to train. Our second model was a residual neural network, which was able to achieve the best training accuracy. Lastly, our final model used transfer learning, first using a pre-trained model to generate an encoding for our image, and finally passing that encoding through three fully connected layer.

1 Introduction

Historically, computers have been incapable of assessing human emotion, and emotion was thought to be outside the capabilities of a machine. Modern advances in deep learning, however, show that this might not be the case. We are interested in building different classifiers that will use deep learning to label the emotions present in a piece of visual art and compare their performance. This challenge is specifically interesting due to the massive complexities found in both art and emotion. Naturally, there is complexity since we are using both computer vision and natural language processing, and since our algorithmic structures are relatively unique (see Methods). We are using a dataset of 80,922 labeled images referenced by Q. You, J. Luo, H. Jin, and J. Yang in the paper *Building a large scale dataset for image emotion recognition: The fine print and the benchmark*. More specifically, the dataset is a folder of CSV files, with each file containing the URL to an image and a label for the following emotions: amusement, anger, awe, contentment, disgust, excitement, fear, and sadness. Each label indicates the most prevalent emotion in the images with an integer – 1 for amusement, 2 for anger, and so on. The input to our algorithm is an image of size (32, 32, 3). We then use any one of our models (Residual Neural Network, Fractal CNN, Transfer Learning) to output a prediction of the most prevalent emotion in the image.

2 Related work

In the paper, "Building a Large Scale Dataset for Image Emotion Recognition: The Fine Print and The Benchmark", the authors describe a dataset of non-facial images that are labeled by their emotions. The researchers fine-tune an existing pre-trained ImageNet reference network[11].They implement the same network, and change the last layer of the neural network from 1000 to 8. For the noisy-fine-tuned-CNN, the researchers incorporated "weak" labeled images in their dataset. This includes a 20,000 contentment images that they had less labels of in comparison to other emotional responses. Lastly, for the fine-tuned CNN, the researchers then used an SVM to train on features from the second to last layer of the pre-trained ImageNet-CNN model. To reduce the dimensionality of the features, they implement PCA. They report that these results are too similar to the output of the pre-trained ImageNet-CNN model[9].

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

These accuracies are not as high as they ideally would be, so the researchers are looking for a better way to analyze the emotions present in images.

Algorithms	Correct Samples	Accuracy
ImageNet-CNN	1120/3490	32.1%
Noisy-Fine-tuned-CNN	1600/3490	45.8%
Fine-tuned-CNN	2034/3490	58.3%

In his paper about spatial neural networks based on fractal algorithms, Kromer explains the benefits of using a fractal CNN in great detail. Some of these advantages include the ability to create relatively deep neural networks without a lot of data as well as a potential increase in accuracy, relative to a regular CNN, as each signal passes many small neural networks on its way through the entire network, therefore acting as a net of nets [3]. One of the weaknesses of Kromer's work, however, is that fractal networks require a large amount of computational power relative to other networks, which is an issue we had when running one of our models.

We also explored other relevant approaches for our task. In the paper "Caffe: Convolutional Architecture for Fast Feature Embedding", the authors list the applicability of feature embeddings. In particular, they mention extracting semantic features from images using a pre-trained network. Using a feature embedding seemed particularly useful for our task. Indeed, there have been many cases with successful embedding to detection or classification, depending on the approach and methods used[11].

Other relevant approaches include residual networks. In his paper, He explains the benefits of these neural networks [1]. Mainly, they allow for identity mapping from one neural network layer to another using "skip" connections. As a result of this, residual neural networks can be deeper than other neural networks without losing accuracy. This is especially useful in cases where there is not a lot of data for a network to learn from, which was the case for our group.

The paper "Deep Residual Learning for Image Recognition" was also extremely valuable in thinking about the benefits and implementation details of our residual learning models. The authors use this framework to facilitate the training of networks that are substantially deeper than previously used models[10]. This paper was pivotal in the choice of one of our models as it provides comprehensive empirical evidence that these types of neural networks are not only easier to optimize but also provide improved accuracy[10].

3 Dataset and Features

As previously mentioned, we are using a dataset of 80,922 labeled images referenced by Q. You, J. Luo, H. Jin, and J. Yang in the paper *Building a large scale dataset for image emotion recognition: The fine print and the benchmark*. In order to parse and clean our data, we wrote a script to read the file contained in the URL, check to see if it is a valid image and download the image with its corresponding label. The labels are amusement, anger, awe, contentment, disgust, excitement, fear, and sadness. We resized the images to (32, 32, 3) while extracting them from the URLs, because we had to fit within the limits of what our memory could handle. We then shuffled the images with their corresponding labels. Our script split this data 90-10-10 into train, validation, and test folders, and our labels were sorted into trainY, devY, and testY vectors, respectively. Out of 80,922 usable images, we retained 67,937 images for train, 8,491 for dev, and 8,492 for test. We decided to split the data this way in order to maximize the size of the training set, given that learning what emotions are present in an image is a relatively complex task that requires learning a lot. Example dataset entries are shown on the following page.

4 Methods

We created three different models for our task.

Fractal Neural Network

We implemented a fractal neural network with softmax output to label the emotions in images. In a fractal net, a simple expansion rule for creating the network will be repeated to generate a neural network structured upon precisely truncated fractals. These networks have sub-paths of different lengths, but there are no residual "skip" connections; every internal signal is transformed by a filter and nonlinearity before being seen by subsequent layers.

Images from Dataset

Label: 0 (amusement) (Left) Original image of size (400x400x3) (Right) Downsized image to 32x32x3

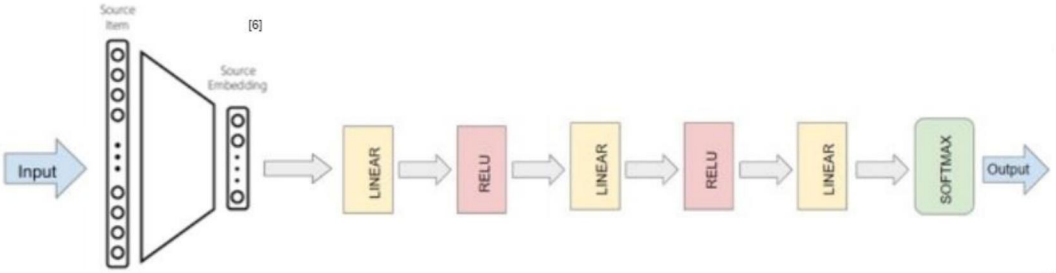
1:angry 2:awe 3:contentment 4:disgust 5:excitement 6:fear 7:sadness

Residual Neural Network

Our second network architecture was the residual neural network architecture. In a residual network, there are "skip" connections which allow information to propagate to deeper layers without being filtered by the weights and biases in a layer. This allows us to create extremely deep networks without losing the identity transformation.

Feature Embedding and Multi-Class Classification Method

The third model we created is slightly more elaborate. We adapted a caption generator[6] pre-trained on the MSCOCO dataset[4] for 2MM iterations. We used the pre-trained weights to input our 32x32x3 image vectors through a CNN and output a feature embedding of size 1024x1. We then trained 90% of our feature embeddings on a 3 layer neural net (https://hub.coursera-notebooks.org/user/mnkglbnnntguifvlabaawn/notebooks/week7/Tensorflow%20Tutorial.ipynb) (Linear->Relu->Linear->Relu->Linear->Softmax) to classify our images according to human emotional response. This gives us the same benefits as transfer learning, while this time allowing our network to do more training.



We implemented a categorical cross-entropy loss. We calculated the loss for each class label, for each observation, then summed the result to get our cost. $\mathcal{L} = -\sum_{c=1}^{M} y_{o,c} log(p_{o,c})$. Here M represents the number of classes (8), log is the natural log, y is 1 if class label 'c' is correct given observation 'o' (otherwise it's 0), and p is the probability that 'o' is of class 'c'. We chose this loss because we want to measure the cost as a function of the misclassifications.

5 Experiments/Results/Discussion

Feature Embedding and Multi-Class Classification Method

The Feature Embedding and Multi-Class Classification approach produced a test accuracy of about 18 percent (see Figure 6). We tuned the following hyperparameters for this model: the learning rate, the mini-batch size and the number of epochs. In Figure 1, we plot the results from our hyperparameter tuning process. Using these graphs, we chose the hyperparameter values shown in the Figure 2. To provide a qualitative understanding of the errors made by the Feature Embedding and Multi-Class

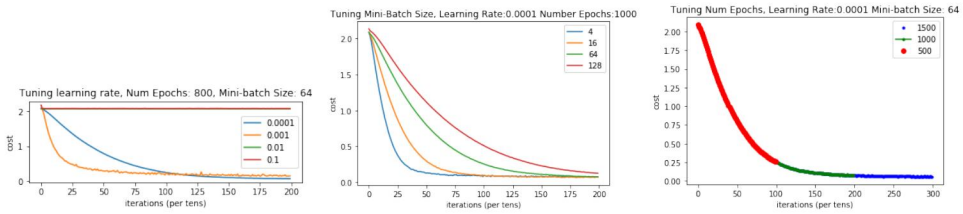


Figure 3: Tuning Learning Rate, the Mini-Batch Size and The number of Epochs

Tuning Hyperparameters

Learning rate (alpha)	0.0001
Number of Epochs	1000
Minibatch Size	64

Figure 4: Optimal Values for the Hyperparameters

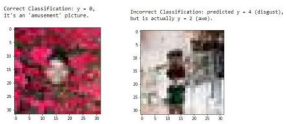


Figure 5: Qualitative Error Analysis

Classification model, we also included an example of an image that was correctly classified by the model as well as an example of a misclassified image (see Figure 3). The first image was correctly classified as amusement, which seems to be the easiest emotion to detect for the model (see Figure 4). As for the misclassified image, the true label is awe, which the model correctly identifies only 15 percent of the time (see Figure 4).

We also noticed that we had a problem with overfitting. In order to mitigate that, we tried L2 regularization and dropout, which led to only slightly improved performance.

Residual Neural Network

The Residual Neural Network performed the best out of all our models with a test accuracy of 24 percent (see Figure 6). In figure 5, we can see three graphs. The first compares validation loss with epoch number, and the second compares train loss with epoch number. We see that validation loss fluctuated a lot, while train loss steadily decreased in a more controlled way. The third graph shows how we decreased the learning rate as epoch number increased.

We tuned many different hyperparameters in the process of training this model. When running the network for the first time, I noticed that each iteration of training took a really long time, so I increased the batch size to 200 for the advantages of vectorization. Next, I ran the model with a small amount of data and noticed that I had a very low training accuracy, so I decided to run the entire dataset through the model to see what lift I would get in terms of accuracy. I saw a large reduction in avoidable bias, but I wanted to see if I could reduce that bias even more. To do so, I tried using different resent architectures, and found that a small, 18 layer network work best after testing 18, 34, 50, 101, and 152 layer networks. Next, I was wondering if increasing the resolution of pictures would have an impact on the accuracy of training, so I changed the pictures to be of size 64x64. I found that neither the training accuracy nor the validation accuracy were improved, and the model just took longer to train. Thus, I switched back to 32x32 images. Then I noticed high variance so

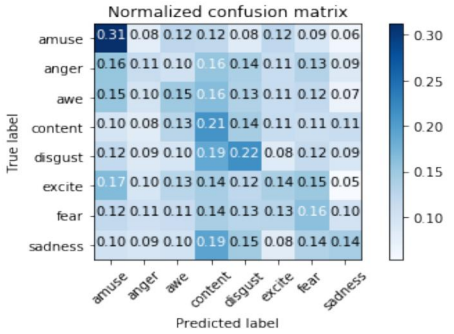


Figure 6: Confusion Matrix

Algorithm	Train Accuracy	Dev Accuracy	Test Accuracy
Residual Neural Network	52%	25%	24%
Fractal Neural Network	13%	13%	13%
Feature Embedding & Multi-Class Classification	46%	20%	18%

Figure 7: Accuracies for all Three Models

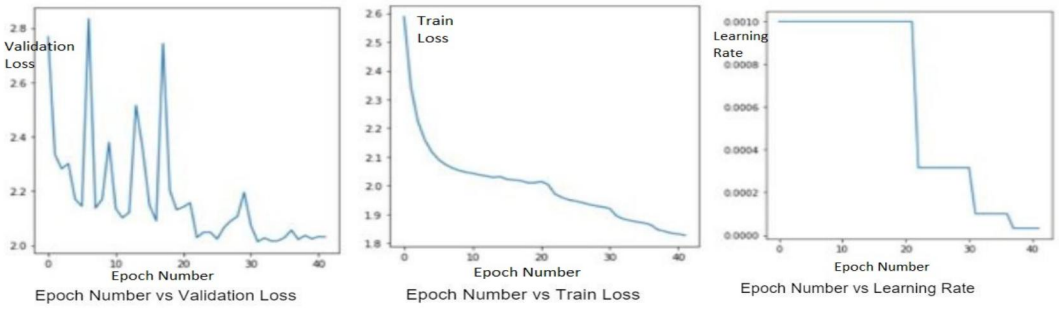


Figure 8: Epoch Number vs Validation Loss, Train Loss and Learning Rate

I added data augmentation. This was very helpful to our variance problem, but I wanted to see if I could do more. In an effort to further decrease variance, I implemented dropout. I found that of 50%, 20%, 10%, and 1% dropout, 1% dropout worked best. Finally, I ran the entire dataset through an 18-layer resnet model with 1% droupout, batch size of 200, 32x32 images, adam optimization, categorical cross-entropy loss, and data augmentation. This was the final model. Our results tell us that our models tend to overfit to the training set for both the Residual Neural Network and the Feature Embedding and Multi-Class Classification (see Figure 6). We tried several different regularization techniques for all of our models, but were still unable to overcome this issue.

6 Conclusion/Future Work

The fractal neural network was not a success. After creating this model, we found that it was extremely computationally expensive and thus infeasible to train. We noticed an exploding gradient problem in the network that we were unable to overcome. Consequently, the fractal neural network never had a high enough accuracy to be considered useful in any way. If we had more computational resources, however, the fractal network might have been feasible.

The residual neural network was much more successful: it achieved the highest test accuracy out of all of our models. We were able to make the residual network deep without sacrificing test accuracy, and as a result we were better able to label images. If we had more time and/or computational resources, it would be interesting to explore more random positions in hyperparameter space to see if we would end up at a better position on the loss function.

Overall, the feature embedding and classification algorithm did not perform as well as we had hoped. We believe that this was mainly due to the loss of information by resizing the images down to 32x32x3, and to the many classes associated with the dataset. We found that using a 32x32x3 downsized image may have been a downsize too large, as we lost a lot of the information in the image, especially trying to classify many emotional responses that are similar. In the end, our algorithm reached an accuracy that was as good as guessing. We believe that this could be handled better by grouping labels that are similar (anger and fear for example). Starting with a classification problem with fewer classes and higher resolution images is what we would do, if we had more time and computational power.

As a whole, our accuracies were lower than would be ideal. We believe this was fundamentally because we did not have enough data for the problem we were trying to solve. The emotions in an image are determined by both the content of the image and the style of the image. Because there are so many options for both content and style, it was difficult for our algorithm to learn based only on the relatively small amount of data it received.

In addition, many would agree that it is possible for one image to elicit several emotions, even though our images are labeled with one emotion each. This might have caused the network to modify parameters that were actually correct during gradient descent. In the future, it would be preferable if the creators of the dataset gave each image a rating of how strongly each emotion appears in each picture. That way, we would ensure that we do not over-penalize partially correct answers.

These reasons also explain why the researchers before us were also unable to achieve high accuracies on the same dataset. Overall, our biggest recommendation to researchers working on this topic would be to create a larger and better-labeled dataset.

7 Contributions

Katherine preprocessed the data: scraping, filtering out bad images, shuffling, splitting, and resizing. She worked on the feature embedding and multi-class classification method. She implemented and adapted an online caption generator open source code[6]. She debugged and extracted feature embeddings for the train, test, dev data, and simplified it, to avoid from computing captions that were not necessary for this project. She input these outputted feature vectors into a three layer neural network, influenced by Coursera Tensorflow programming assignment (https://hub.coursera-notebooks.org/user/mnkglbnnntguifvlabaawn/notebooks/week7/Tensorflow%20Tutorial.ipynb). She hyper tuned parameters of learning rate, number of epochs, and minibatch size. She provided graphs of her results of hyper parameter versus cost and training accuracy. She trained this three layer CNN, hyper tuned on the dev set, and tested. She worked with debugging these models, as well as producing figures (as seen on the poster) to visualize results of the model. Katherine also worked with Noah to set up AWS to run the models, with the creation of an EC2 instance.

Noah resized the data to 32x32x3. He created the fractal neural network with some help from Katherine and created the residual neural network completely on his own, adapting several open source infrastructures to meet the project's needs. He also did 100% of the training for both the fractal and the residual neural network. In the fractal net, he tuned minibatch size, learning rate, and added data augmentation. In the residual network, he tuned minibatch size, learning rate, image resolution, dropout, added regularization, and added data augmentation. He also created graphs of the learning rate and performance of both networks as they trained. Finally, Noah, with help from Katherine, figured out the details of AWS and set up our team's EC2 instance.

Early on, Hasna thoroughly researched different potential approaches for using transfer learning, which was useful for the multi-class classification model. Hasna then used that knowledge to create the feature embedding and multi-class classification model. Along with Katherine, she debugged and extracted feature embeddings for the train, test, dev data and used these feature vectors as an input to the three layer neural network. Katherine and Hasna trained this three layer CNN and tuned the hyperparameters on the dev set. Hasna also created the confusion matrices for the true vs predicted labels based on the results. She also wrote the milestone, the Experiments/Results/Discussion section, as well as the related work section in which she used the research she did earlier on. She also handled the logistical aspect of the poster session such as printing the poster and setting up for the session.

References

[0] Chollet, François. Keras, code found on keras.io

[1] He, Kaiming, et al. "Identity Mappings in Deep Residual Networks." *[Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field*, American Physical Society, 25 July 2016, arxiv.org/abs/1603.05027.

[2] Jay, Prakash. "Image Classification Architectures Review – Prakash Jay – Medium." *Medium.com*, Medium, 19 July 2018, medium.com/@14prakash/image-classification-architectures-review-d8b95075998f.

[3] Kromer, Thomas. "Fractal Neural Networks." Introduction to Fractal Geometry, www.fractal.org/Life-Science-Technology/Publications/Fractal-Neural-Networks.htm.

[4] Lin, T., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. and Dollár, P. (2018). *Microsoft COCO: Common Objects in Context.* [online] Arxiv.org. Available at: https://arxiv.org/abs/1405.0312 [Accessed 12 Dec. 2018].

[5]Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia,Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster,Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens,Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng.TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[6] Murray, C. (2018). *Building an image caption generator with Deep Learning in Tensorflow.* [online] freeCodeCamp.org. Available at: https://medium.freecodecamp.org/building-an-image-caption-generator-with-deep-learning-in-tensorflow-a142722e9b1f [Accessed 12 Dec. 2018].

[7] Raghakot. "Raghakot/Keras-Resnet." GitHub, 6 July 2017, github.com/raghakot/keras-resnet.

[8] Snf. "Snf/Keras-Fractalnet." *GitHub*, 17 Sept. 2017, github.com/snf/keras-fractalnet/.

[9] You, Q., Luo, J., Jin, H. and Yang, J. (2018). *Building a Large Scale Dataset for Image Emotion Recognition: The Fine Print and The Benchmark.* [online] Arxiv.org. Available at: https://arxiv.org/abs/1605.02677 [Accessed 12 Dec. 2018].

[10] Zhang, Xiangyu, et al. "Deep Residual Learning for Image Recognition." *[Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field*, American Physical Society, 10 Dec. 2015, arxiv.org/abs/1512.03385.

[11] Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. *Caffe: Convolutional architecture for fast feature embedding.* arXiv preprint arXiv:1408.5093.