Categorizing Facial Expressions with Deep Learning

Jared Diehl

May 2021

Contents

1 Introduction

We have always had the innate ability to recognize faces. Now computers could be capable of doing the same. We are emotional creatures, and we express emotions through our facial appearances. Suppose we can build a machine learning model to recognize just a handful of human emotions. In that case, this could lead to a more sophisticated computer recognition system. This project aims to do just that: develop a machine learning model

to categorize several facial expressions: angry, astonished, neutral, happy, and sad. The development of the model is documented in this report, along with visualizations to aid understanding. This process is separated into several distinct phases to help learn how to build, train, and evaluate machine learning models. First, a variety of models will be tested to find which models overfit all the data. Then a more efficient will be built and evaluated on the validation and test sets. The techniques used to reduce overfitting will include data augmentation and model regularization. Data augmentation artificially creates more diverse data. Model regularization will reduce complexities within the model. In the end, transfer learning will be explored to find if a pretrained model learns faster and makes better predictions. Ultimately, this project can not work without high-quality data, so several hundred selfies were selected carefully and cleaned. The cleaning process involved preprocessing and normalizing the images. Google Colab was used to develop this project at a rapid speed across several months. This project will serve as an example of how to build, train, and evaluate a convolutional neural networking model capable of categorizing 5 unique facial expressions.

Link to Project: https://github.com/jaredible/CS4390-Project

2 Data Preparation

2.1 Image Selection

A total of 1000 images of five unique facial expressions were randomly collected of my face. The images were selected such that a single class is present within the image. They were chosen so that each are relatively unique. Each facial expression was captured in groups of nine with random a face direction and lighting condition. Images that were blurry or not in frame were discarded.

2.2 Preprocessing Data

The images were cleaned to ensure that the facial features are clearly visible, as they have many details. All were adjusted to a 256 x 256 grayscale image and shuffled. They should now be free of errors and bias, so it is ready to be used as a dataset for a model.

2.3 Data Distribution and Visualization

The distribution of the data is defined in Table 1, along with the visualization of the data in Figure 1. No data imbalance exists in the dataset as there are 200 images for each class.

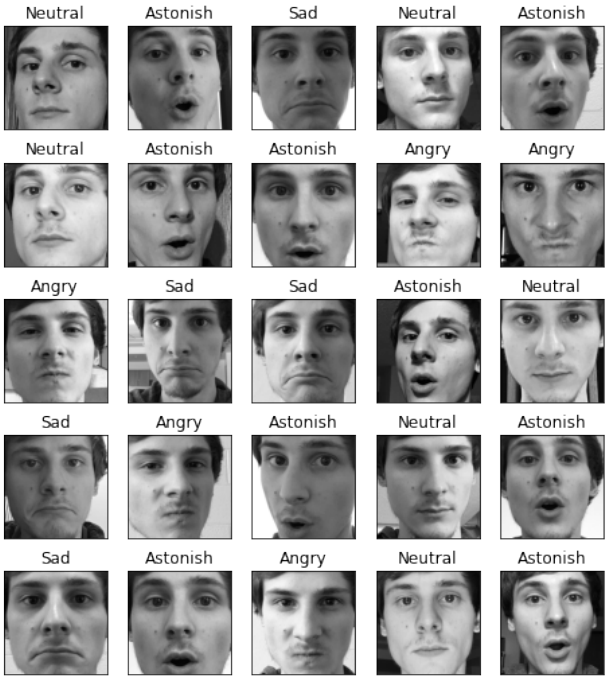

Figure 1: Visualization of Data Sample

Angry	200
Astonish	200
Neutral	200
Happy	200
Sad	200

Table 1: Data Distribution

2.4 Data Normalization

The images were normalized along their channel using the rescaling technique as seen in (1), where $Y_{max} = 255$. This is to prevent any potential skewing. Thus, the model should converge to the optimal trainable parameters much more quickly.

$$Y_{new} = \frac{Y}{Y_{max}} \tag{1}$$

3 Building an Overfit Model

A model is a complex mathematical structure used to make predictions from data. A series of convolutional neural network models were tested to find the one that made the best predictions. This type of model is called an overfit model, and is the first step to creating a generalized neural network model.

Overfitting refers to the scenario where a machine learning model cannot generalize or fit well on unseen data. [2] There are multiple ways to overfit the data. A series of 4 models were trained using all the data. Evaluation statistics of all the tested models can be seen in Table 2.

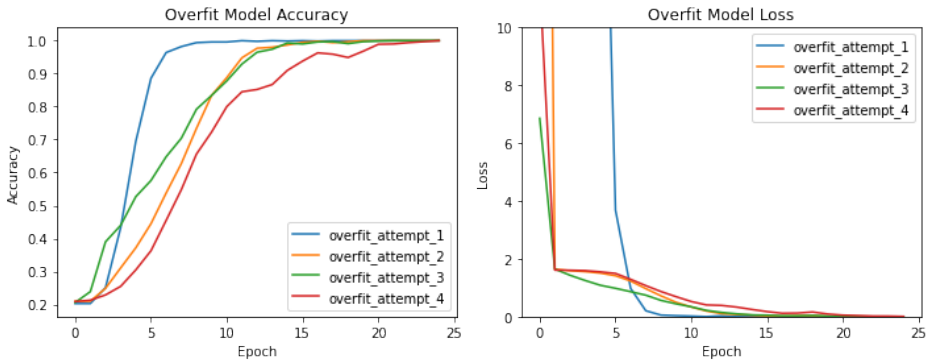

Figure 2: Learning Curves of Overfit Models on All Data

All models use the softmax activation function for their last layer and relu activation for all other layers. Each were trained in 25 epochs and with a batch size of 64. Making the network denser and wider seemed to make the model have higher performance. To speed up the training process, the early-stopping technique was used and slightly reduced the total training time for the models.

	Loss	**Accuracy**
overfit_attempt_1	2.7045e-07	100.00%
overfit_attempt_2	9.2153e-04	100.00%
overfit_attempt_3	0.0017	100.00%
overfit_attempt_4	0.0028	100.00%

Table 2: Evaluation of Overfit Model Attempts

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 256, 256, 8)       80
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 128, 128, 8)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 16)      1168
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 64, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 32)        4640
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 256)               2097408
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 1285
=================================================================
Total params: 2,196,933
Trainable params: 2,196,933
Non-trainable params: 0
```

Figure 3: Best Overfit Model Architecture

The model architecture that best overfit the data can be seen in Figure 3, and is the largest of the 4. Each convolutional layer's filters decrease the deeper into the network. This means that the first convolutional layer is capable of distinguishing many small differences between facial features. At first, the first dense layer had 128 neurons, but changing it to 256 greatly increased training speed.

4 Splitting and Evaluating on Test Set

The 1000 images were separated into sets as described in Table 3. All overfit attempt models were retrained using the validation set and evaluated on the test set.

	Angry	**Astonish**	**Neutral**	**Happy**	**Sad**
Train	120	120	120	120	120
Validation	40	40	40	40	40
Test	40	40	40	40	40

Table 3: Set Distribution

To increase development speed, the training would finalize if the validation loss did not improve in 30 epochs. Essentially, this minimized the validation loss just enough so the unnecessary training time is avoided.

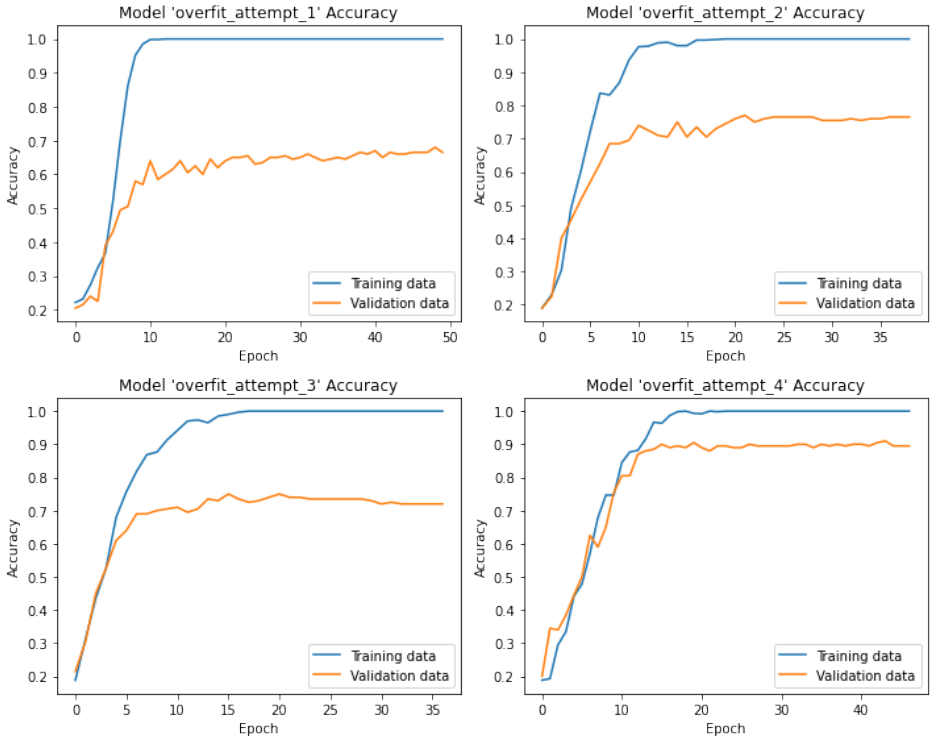

Figure 4: Learning Curves of Models

	Loss	Accuracy
overfit_attempt_1	0.9533	63.50%
overfit_attempt_2	1.2943	75.50%
overfit_attempt_3	1.4482	76.00%
overfit_attempt_4	0.2911	95.50%

Table 4: Evaluation of Models on Test Set

5 Effects of Data Augmentation

Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks. [1]. As can be seen in Figure 5, each model learned at a steady pace.

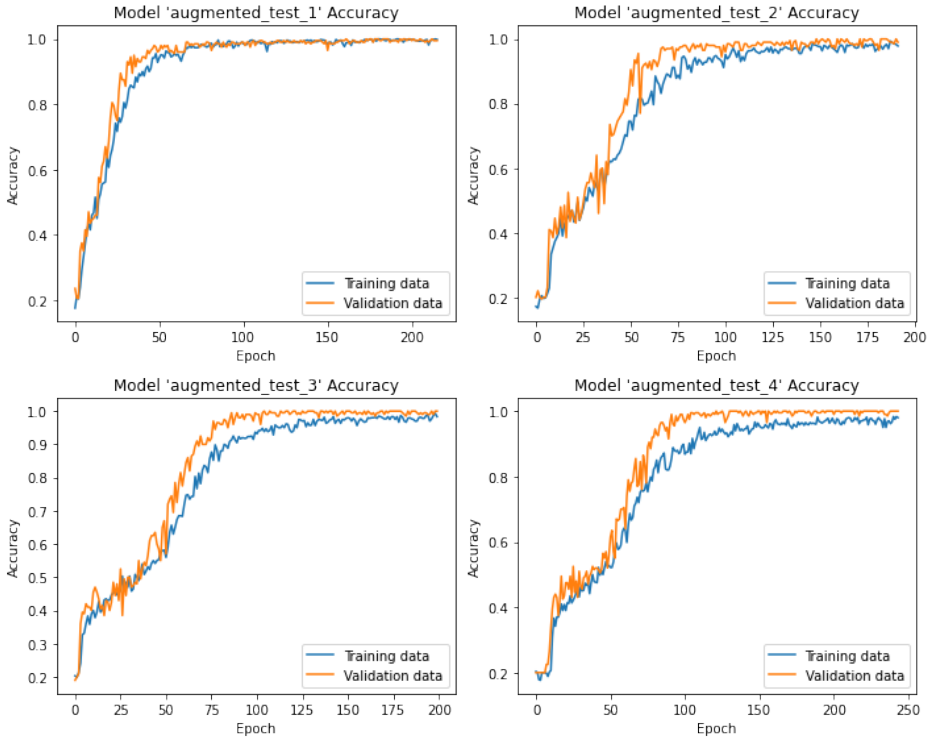

Figure 5: Learning Curves of Overfit Model with Augmentated Data

The best performing model trained with augmented data is **augmented_test_4**. Although, some images were badly augmented so a safer strategy needed to be used. The augmentation parameters used in **augmented_test_2** sufficed because it did not learn too fast or too slow. As can be seen in Table 5, all attempts made perfect predictions, which helps them generalize.

	Loss	**Accuracy**	**Variation**
augmented_test_1	0.0044	100.00%	smallest
augmented_test_2	0.0111	100.00%	smaller
augmented_test_3	0.0037	100.00%	small
augmented_test_4	0.0019	100.00%	small

Table 5: Evaluations of Overfit Model with Augmentated Data

6 Effects of Regularization

Regularization helps the model reduce complexity by penalizing it during the training process. This reduces overfitting and is a solution to making the model generalize. The

chosen regularization techniques were L2 regularization, dropout, and batch normalization. Each model test used a single technique except the last one which used all 3. As can be seen in Figure 6, each model learned at a steady pace. Ultimately, the regularization techniques used did not greatly improve upon the model prior to regularization.

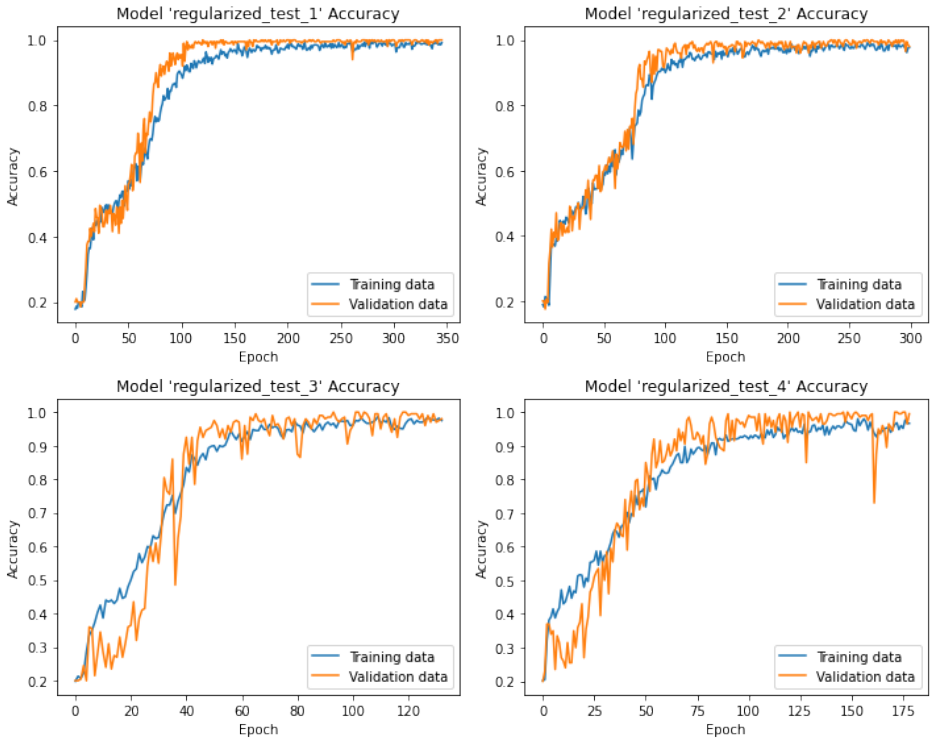

Figure 6: Learning Curves of Overfit Model with Regularization Techniques

The best performing model is **regularized_test_3** as it was capable of learning the fastest. With more time training, this model could be perfect. Table 6 shows the results and the learning curves can be seen in Figure 6. Batch normalization caused the largest improvement, whereas L2 regularization and dropout had the smallest. The results of batch normalization

	Loss	**Accuracy**	**Technique**
regularized_test_1	0.0167	99.50%	L2 Regularization (first conv layer)
regularized_test_2	0.0325	99.00%	Dropout (after last maxpool)
regularized_test_3	0.0119	99.50%	Batch Normalization (after last conv layer)
regularized_test_4	0.0608	98.00%	All 3

Table 6: Evaluations of Overfit Model with Regularization Techniques

7 Using Pretrained Architectures

To study the effects of transfer learning, a series of models pretrained on the ImageNet dataset were tested. The convolutional base of each pretrained model was frozen and a new dense network from the overfit model was connected. As can be seen in Figure 7, all models except for ResNet50 learned well enough.

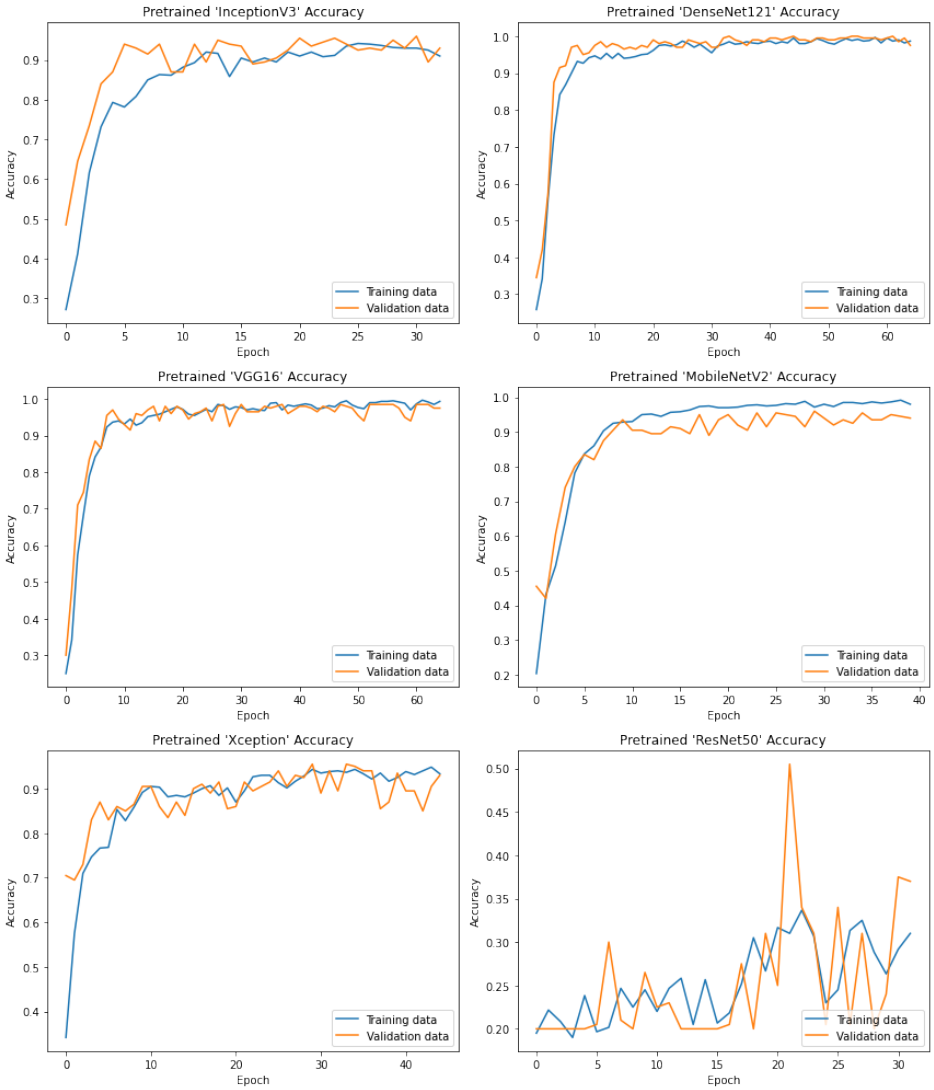

Figure 7: Learning Curves of Pretrained Models

The fastest learning model was **Xception** and the most accurate model was **VGG16**, as can seen in Table 7. This means that both models' convolutional bases had enough

knowledge to classify the facial features. The models were trained to until the validation loss was minimized within 10 epochs.

	Loss	Accuracy
InceptionV3	0.26	90.50%
DenseNet121	0.06	98.00%
VGG16	0.03	99.50%
MobileNetV2	0.09	96.50%
Xception	0.11	96.50%
ResNet50	1.48	40.50%

Table 7: Evaluations of Pretrained Models

8 Conclusion

The goal of this several-month-long project was met as a generalized machine learning model was found. This project now serves as a complete example of building, training, and evaluating a convolutional neural networking model capable of categorizing 5 unique facial expressions. The data collected was sufficient enough to find this model and test. With even more data, a more generalized model could be constructed. The learning curves and evaluations of the models prove this finding across all the development phases. The hyperparameters were found that yielded a generalized predictive model across the test set. Building an overfit model was also quick to find and evaluate on the test set with the help of the early-stopping technique. The effects of data augmentation and model regularization significantly reduced overfitting. The optimal augmentation parameters were found to ones that did not add too much diversity. Over-diversification caused bad data to be produced, such as incorrectly cropped or dark images. The typical regularization techniques used were sufficient enough to improve training speed and reduce complexity slightly. Overall, model regularization did not significantly improve the development process. This could be because very few tests were conducted. Also, pretrained architectures showed that transfer learning is highly efficient for building models with similar classification purposes. Ultimately, this project is an example of when a pretrained network would reduce overall development time and costs. This project can help facilitate understanding of how convolutional networks learn and may serve as inspiration to go beyond just these types of models.

References

[1] Daniel Ho, Eric Liang, and Richard Liaw. *1000x Faster Data Augmentation*. URL: https://bair.berkeley.edu/blog/2019/06/07/data_aug/. (accessed: 06.07.2019).

[2] Mayank Tripathi. *Underfitting and Overfitting in Machine Learning*. URL: https://datascience.foundation/sciencewhitepaper/underfitting-and-overfitting-in-machine-learning. (accessed: 03.25.2021).