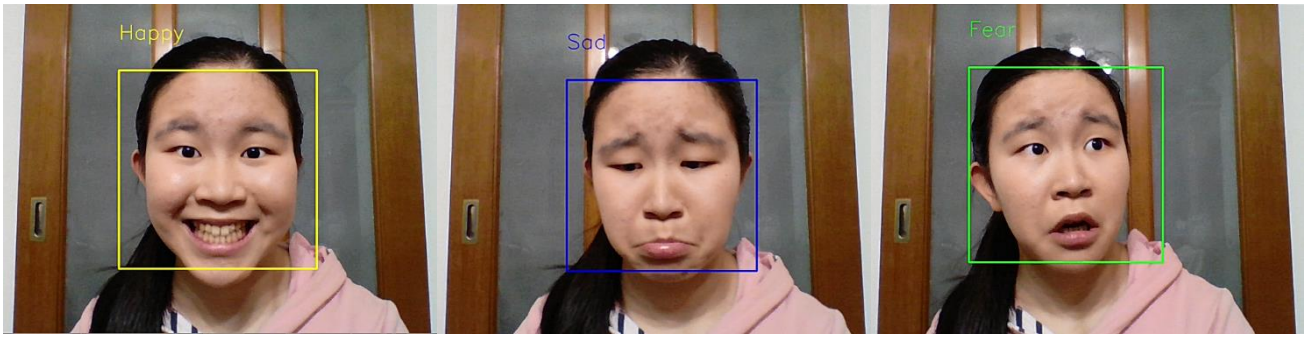
Emotion Recognition Model

Chi-Feng Wang



Title image: A demonstration of the model

Github Link: https://github.com/reinaw1012/Emotion-Recognition

Overview:

During my internship, as I was surveying different face detection models, I came across an emotion recognition model by the B-IT-BOTS Robotics Team, which achieved a 66% accuracy [1]. For a classification neural network that has 7 categories, 66% is quite good. This model uses a Haar-feature based cascade classifier to detect faces from each frame of videos, then passes the faces through a neural network to recognize emotions. Using this model as a baseline, I built my own emotion recognition neural network with Python, using the Keras and OpenCV libraries, and trained it with the FER-2013 emotion image database.

Substituting Haar Classifier for MTCNN:

Before I built the model, I wanted to experiment with the Haar-feature based cascade classifier. Haar features are kernels that can be used to detect lines and edges, like the one in Figure 1. As it can be seen, Haar features can be used to detect facial landmarks, such as the shadow of an eye.

Figure 1

However, Haar-feature based cascade classifiers aren't neural networks. Wanting to create an entirely neural-network based emotion recognition model, I experimented with substituting the face detection neural network I surveyed during my internship, the MTCNN face detection algorithm [2], for the Open CV Haar-feature based cascade classifier [3]. After running both models, a few observations were made:

1. The OpenCV Haar classifier was significantly faster. After switching it to the MTCNN detector, the video started to lag.
2. The MTCNN detector was able to detect a larger variety of faces, whether it be tilted, partially turned away from the camera, or partially obscured. The OpenCV Haar-based classifier could only recognize full front-facing faces.
3. The emotion recognition network could only accurately recognize different emotions on full front-facing faces. Hence, even if the MTCNN detector allowed us to draw a bounding box around partially obscured faces, the program couldn't accurately recognize the emotion on the face.

With consideration of the speed, I decided to switch back to the Haar Classifier.

Building my own model:

I experimented with several models to test their accuracies. The first convolutional neural network (CNN) model I made, as seen in Figure 2, has 7 convolutional layers and 2 dropout layers. It is a relatively small network, with the maximum width of each layer being 64. It achieved an accuracy of 54%. To test if a larger network would result in a significantly higher accuracy, I built a second network, seen in Figure 3. With 10 convolutional layers, 5 dropout layers, and 2 fully-connected layers, it was able to achieve an accuracy of 58% but required almost twice the amount of training parameters.

To reduce the number of training parameters, I created a third model, seen in Figure 4. This model only has 7 convolutional layers, 3 dropout layers, and 2 fully-connected layers, but is a wider model than the first. This model was able to achieve an accuracy of 56%. Ultimately, I came across the FeatEX model [4], originally designed for the Cohn-Kanade and MMI Facial Expression Database (CK) as seen in Figure 5. Because the FER-2013 and the CK database has differently size images, I had to change the parameters of every layer, to adapt it for the emotion recognition model. The FeatEX model utilizes different data pathways, making it more accurate than my previous linearly-connected models. Not only did this model achieve a 63% accuracy, it also reduced the number of training parameters down to half of the original model.

Generating Hard Data:

To improve accuracy, hard data mining was conducted—that is, the network was further trained on the images it predicted wrong. I decided to conduct hard data mining on the third model, because

it was small and simple enough to clearly distinguish the results, as compared to the larger second model or the more complex FeatEX model. In order to do so, the network was first trained, then made to predict the emotions of all the training data. The false predictions were saved, and cropped and rotated by the Keras ImageDataGenerator to produce a wider variety of input data. After a second round of training was conducted on its false predictions, its training accuracy rose to 80% but its validation accuracy fell to 20%. Finally, after a third training session with the full database, the validation accuracy rose to 66%.

Removing Disgust and Anger:

To recognize the full dataset the network had to learn 7 emotions. Wanting to see the effects of reducing the variation of emotions, I decided to reduce the dataset to 5 emotions. I checked the number of images for each emotion in the dataset and reviewed the images of each emotion. In the end, I removed disgust—because there were only 436 training images as opposed to 3000+ for all the other emotions. I also removed anger, because the images of anger looked very similar to those of sadness and fear, which could confuse the network.

The FeatEX network was retrained without disgust and anger, and achieved a 70% accuracy, compared to 66% of the B-IT-BOTS Robotics Team model.

Towards Data Science:

Many online resources have supported me throughout the process of making this model. Therefore, I decided to record what I learned on the Towards Data Science publication, for others interested in building neural networks.

Webpage: https://towardsdatascience.com/different-ways-of-improving-training-accuracy-c526db15a5b2

References:

[1] Arriaga, Octavio et al, (2016), Github repository, https://github.com/oarriaga/face_classification

[2] Zhang, Kaipeng et al. “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks.” IEEE Signal Processing Letters 23 (2016): 1499-1503.

[3] OpenCV: Cascade Classifier, A Documentation of the Haar feature-based cascade classifer, https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[4] Burkert, Peter et al. “DeXpression: Deep Convolutional Neural Network for Expression Recognition.” CoRR abs/1509.05371 (2015): n. pag.

Models:

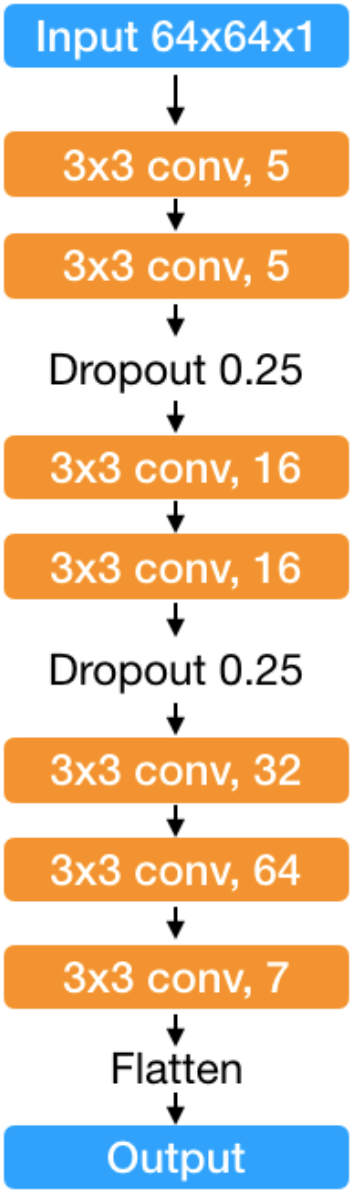


Figure 3: First model

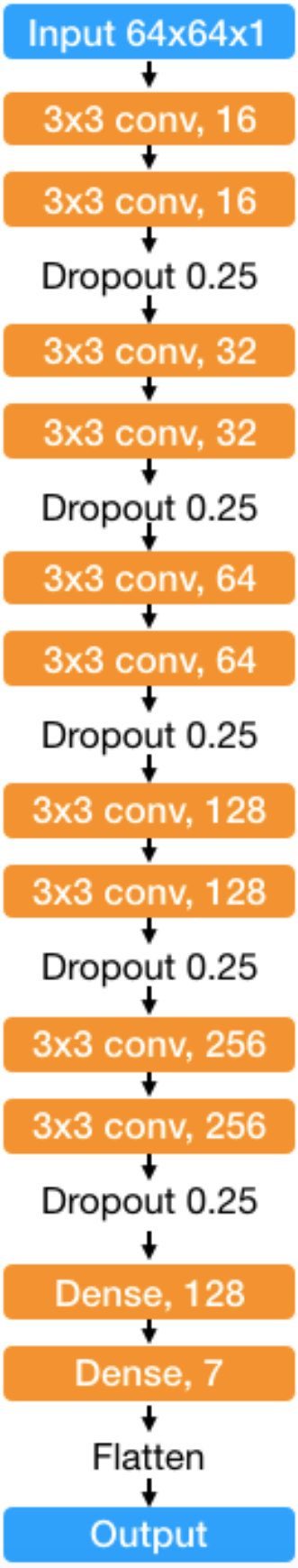


Figure 2: Second model

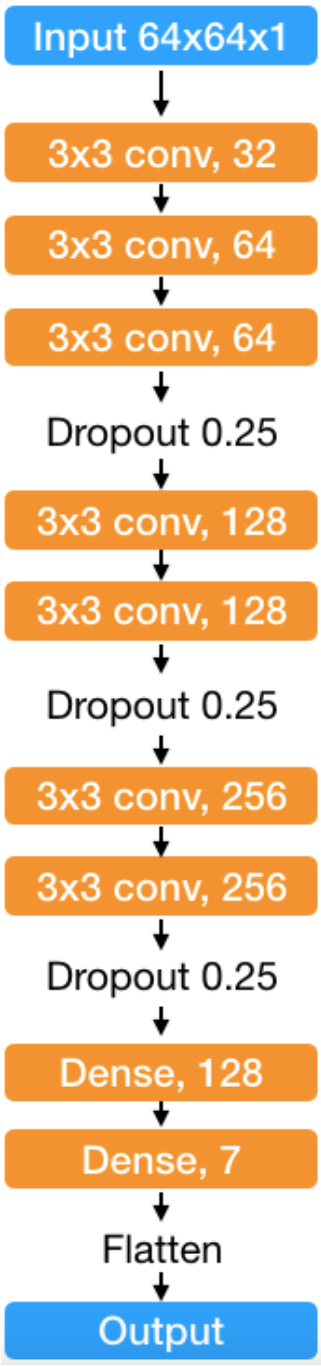


Figure 4:Third model

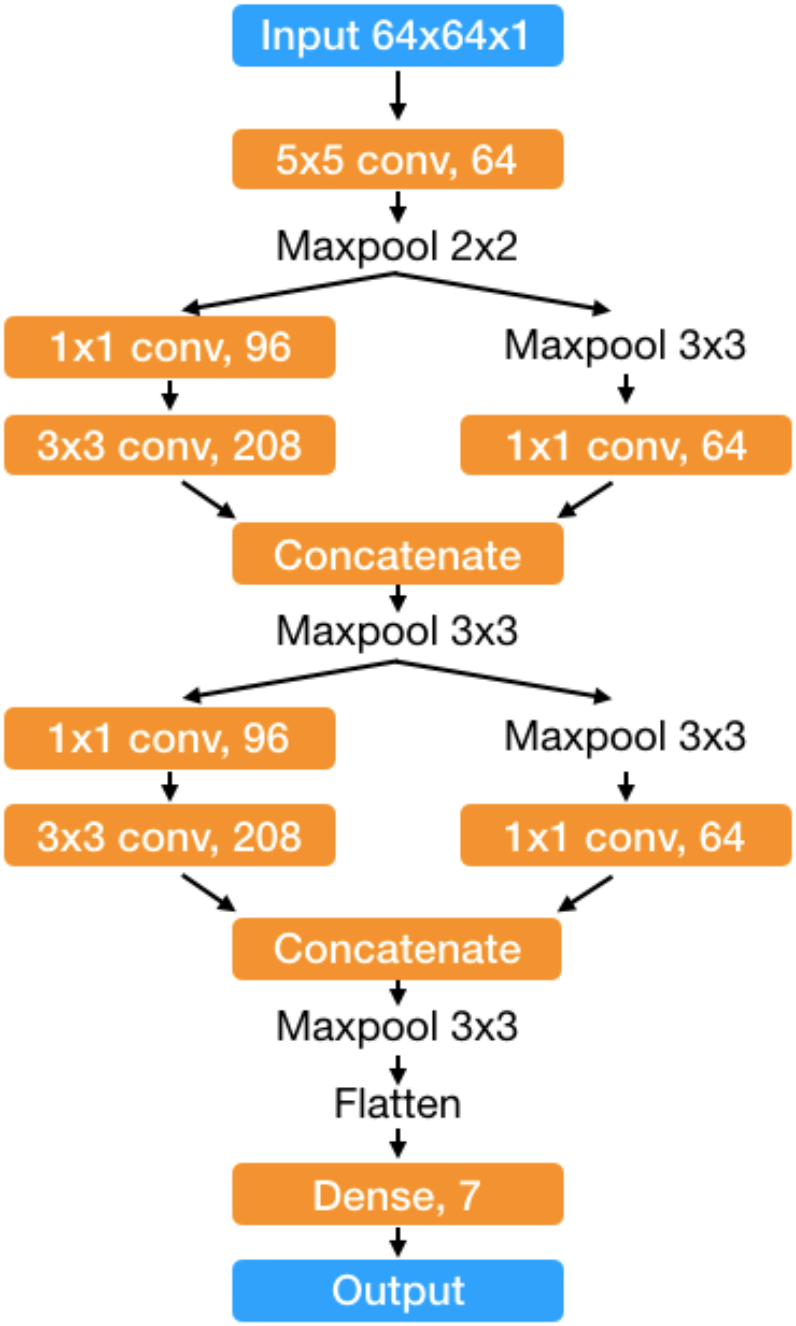


Figure 5: FeatEX Model