Emotion Recognition Neural Network

Final Project Paper for DATA622: Machine Learning & Big Data

Daniel Dittenhafer

December 13, 2016

Submit your final project, which includes the source code of your best model, plus a short (3-4 page) paper summarizing your work. Provide:

- an overview of the problem;
- why it's challenging;
- what techniques you used;
- comparison of performance;
- generalizable lessons learned/takeaways.

1 Overview

As part of the course requirements for Machine Learning & Big Data (DATA622), the problem of developing a machine learning model capable of classifying human face images into various emotions was presented. Specifically, the goal was to design and train a neural network to recognize the following 8 emotions and emit a probability of each of the emotion classes given the input 2 dimensional (2D) image.

1. anger
2. contempt
3. disgust
4. fear
5. happiness
6. neutral
7. sadness
8. surprise

Although this problem is not new to machine learning and solutions already exist (Microsoft Corporation, 2016), the complexities of the mathematics alone are challenging. This, combined with the application of a variety of neural network techniques, make for many unique solutions. Much of the academic theory for the neural network was derived from the text by Ian Goodfellow, et al, ***Deep Learning*** (Goodfellow, Bengio, and Courville, 2016).

2 Techniques

The DATA622 class as a whole, under Professor Rowe's guidance, agreed to a general approach using the *Labeled Faces in the Wild* face database as a basis for training data (Huang, Ramesh, Berg, and Learned-Miller, 2007). We then labeled each of the included images using the Microsoft Emotion API. This was considered the "gold standard" for our training purposes and shared amongst the class as well as publicly in the facial_expressions repository on GitHub (Rowe, 2016).

In order to increase the size of our training and test data sets as well as apply regularization via data, each member of the class created ten image transformations using a variety of APIs and shared the transformations with the class as options for each of us when augmenting our data sets. The following code shows an example of a transformation applied during training (Joshi, 2015):

```
def cvBlurMotion1(img):

    size = 15
    kernel_motion_blur = np.zeros((size, size))
    kernel_motion_blur[int((size - 1) / 2), :] = np.ones(size)
    kernel_motion_blur = kernel_motion_blur / size

    img2 = cv2.filter2D(img, -1, kernel_motion_blur)
    return img2
```

The function, above, along with all code developed as part of this project can be found in the emotional-faces repository on GitHub.

The initial model was based on a model shared on Kaggle as part of the digit recognition competition (Majumdar, 2016). In general, it used 3 convolutional layers plus 2 dense layers with rectified linear unit activation and a softmax final activation for the class probablities. This initial model was adjusted in minor ways, with followup training and validation to measure performance.

The final model was very similar to the initial model in many ways. It still relied on convolutional layers at the input and hidden layers with 2 dense layers on the end, but had a forth convolutional hidden layer internally.

```
def emotion_model_jh_v5(outputClasses, input_shape=(3, 150, 150), verbose=False):
    model = Sequential()
    model.add(Convolution2D(32, 8, 8, input_shape=input_shape))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Convolution2D(32, 5, 5))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Convolution2D(64, 3, 3))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Convolution2D(64, 2, 2))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(64))
    model.add(Activation('relu'))
    #model.add(Dropout(0.4))
    model.add(Dense(outputClasses))
    model.add(Activation('softmax'))

    if verbose:
        print (model.summary())

    model.compile(loss='binary_crossentropy',
                  optimizer='rmsprop',
                  metrics=['accuracy'])

    return model
```

As you can see, the structure is approaching "deep" with the 4 hidden layers. Importantly, the convolutional window begins larger (8x8) on the 150x150 image, and gradually narrows through the remaining 3 convolutional layers.

Binary cross entropy was used for the loss function, and Root Mean Square Propagation (RMSProp) was used for the optimization. RMSProp was used for its ability to consistently descend toward the minimum though admittedly not the fastest to arrive (Reid, 2014). Kind of a tortise approach in a tortise vs hare annalogy.

Convolutional 2D, hyper params

Max Pooling

Drop out

optimization function

3 Performance

TBD

4 Conclusions

TBD

5 References

Goodfellow, I., Y. Bengio and A. Courville. Deep Learning. Sep. 2016. URL: http://deeplearningbook.org.

Huang, G., M. Ramesh, T. Berg and E. Learned-Miller. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. Tech. rep. University of Massachusetts, Amherst, Oct. 2007. URL: http://vis-www.cs.umass.edu/lfw/.

Joshi, P. OpenCV with Python By Example. Sep. 2015. URL: https://www.packtpub.com/mapt/book/application-development/9781785283932/2/ch02lvl1sec23/Embossing.

Majumdar, S. Deep Convolutional Network using Keras. Apr. 2016. URL: https://www.kaggle.com/somshubramajumdar/digit-recognizer/deep-convolutional-network-using-keras.

Microsoft Corporation. Emotion API. Dec. 2016. URL: https://www.microsoft.com/cognitive-services/en-us/emotion-api.

Reid, S. 10 misconceptions about Neural Networks. May. 2014. URL: http://www.turingfinance.com/misconceptions-about-neural-networks/#algo.

Rowe, B. facial_expressions. Dec. 2016. URL: https://github.com/muxspace/facial_expressions.