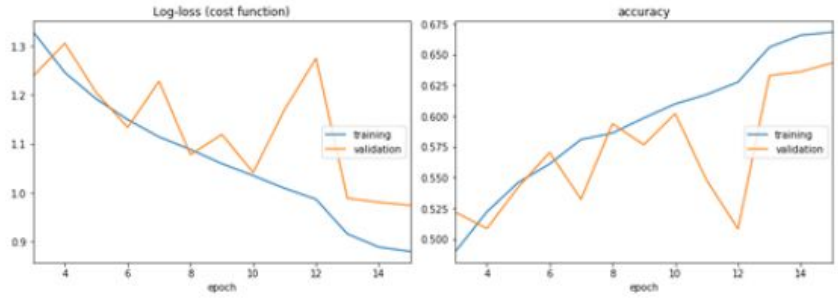
Deliverable #2

The machine learning problem that I am solving is the mapping of a photo containing a person's face to an emoji corresponding to that person's facial expression. For this project, I am focusing on 6 specific expressions: angry, disgust, fear, happy, sad, surprise, and neutral. After classifying the expression, the web app will superimpose an emoji on top of the photo input as output.

During the course of my project, I decided to change the dataset that I am working with. The new dataset that I am working with is also from Kaggle, but from a different competition: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data. There were a few reasons why I decided to change the dataset that I am working with. Firstly, the images from the dataset I proposed in deliverable #1 were quite ambiguous: when I was looking through the photos, it was difficult for me to manually determine what expression I would classify the person in the image to have. Furthermore, the images and the labels were stored separately in different folders with different orders, and transforming the images to a string of integers containing grayscale values and matching the images to the correct label was challenging given the volume of the dataset. The new dataset does not possess neither of these two problems: the data is processed so that all the images are represented in one csv file as 48x48 pixel greyscale images (with greyscale values stored a string), and when I displayed some of these images, it was clear what expression each image should be classified as.

I have decided to continue with the CNN model that I proposed in Deliverable 1. Because I have never worked with the Keras library or CNNs before, for this deliverable, I enrolled in a Coursera course on facial expression classification which works with CNNs and Keras (recommended to me by my exec buddy Ying) (https://www.coursera.org/learn/facial-expression-recognition-keras/home/welcome) and worked with the course to create a CNN from start to finish that can recognize facial expressions. The model that I built with the course first fits the input through four 2D convolutional layers, and then through two fully connected layers. For each of the 6 layers, batch-normalization and relu non-linearity was enabled. The Adam optimizer (a stochastic gradient descent method) was used as it was computationally efficient. 15 epochs were used for the data which had approximately 6700 examples. The model's accuracy was monitored throughout the 15 epochs (during which it increased to and plateaued at around 64%), and model weights were saved to "model_weights.h5". The accuracy of my training and test (validation) sets through each of the 15 epochs was graphed:




```
Log-loss (cost function):
training   (min:    0.880, max:    1.799, cur:    0.880)
validation (min:    0.974, max:    1.541, cur:    0.974)

accuracy:
training   (min:    0.310, max:    0.668, cur:    0.668)
validation (min:    0.386, max:    0.643, cur:    0.643)

Epoch 00015: saving model to model_weights.h5
448/448 [==============================] - 27s 60ms/step - loss: 0.8801 - accuracy: 0.6680 - val_loss: 0.9743 - val
_accuracy: 0.6430
```

There are many next steps that I will take during the next two weeks. Firstly, I will experiment with different CNN architectures (such as changing the number of conv blocks and connected layers, implementing other structures from Keras, etc.) using the model that I built with the course as a starting point/guideline. Because the architecture of the neural network was determined by the instructor of the course, my next step is to design my own CNN architecture now that I have a better understanding of the Keras library and CNNs. I will also experiment with different hypervariables such as learning rate, learning rate adjustment, epochs, 2D conv block filters, etc. to improve the accuracy of the model. Furthermore, I will implement the last part of my project, which is to replace the face in the image with an emoji.