Week 2

Creating Convolutional Neural Network in JavaScript

- This week would process images in browser
- Also write browser app to use convolutional neural network
- Before doing that, the first thing to do is to create the model and the layers

```
model = tf.sequential();
model.add(tf.layers.conv2d({inputShape: [28, 28, 1], kernelSize: 3, filters: 8,
  activation: 'relu'})
model.add(tf.layers.maxPooling2d({poolSize: [2, 2]}));
model.add(tf.layers.conv2d({kernelSize: 3, filters: 16, activation: 'relu'});
model.add(tf.layers.maxPooling2d({poolSize: [2, 2]}));
model.add(tf.layers.flatten());
model.add(tf.layers.dense({units: 128, activation: 'relu'}));
model.add(tf.layers.dense({units: 10, activation: 'softmax'});
```

- To compile and fit Model

```
model.compile(
  { optimizer: tf.train.adam(),
    loss: 'categoricalCrossentropy',
    metrics: 'acc'}
);

model.fit(trainXs, trainYs, {
  batchSize: BATCH_SIZE,
  validationData: [testXs, testYs],
  epochs: 20,
  shuffle: true,
  callbacks: fitCallbacks
});
```

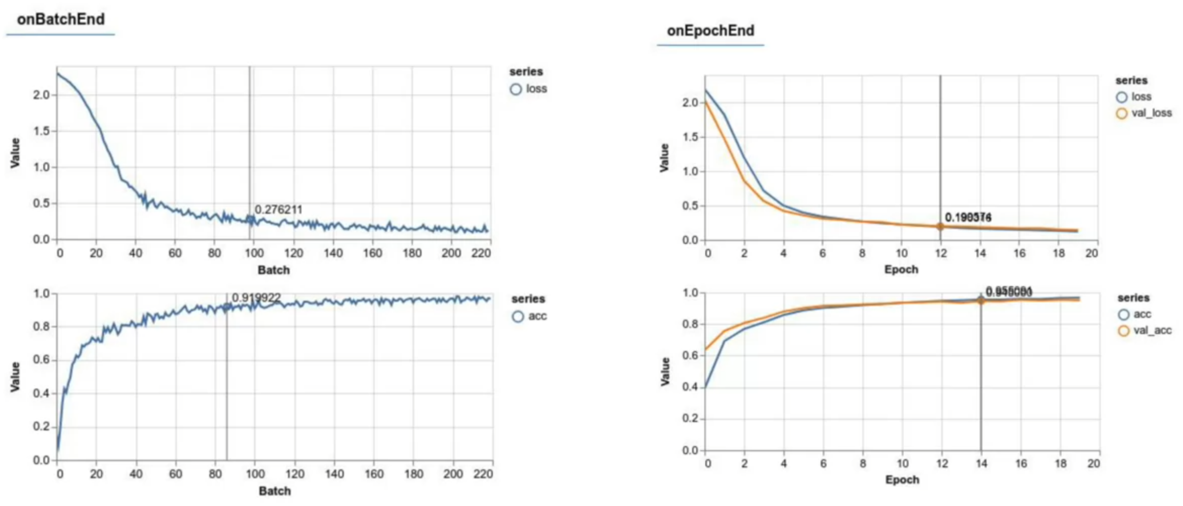
Visualizing Training Process

- Include the library

```html
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs-vis"></script>
```

- To use TF-visualization

```javascript
const metrics = ['loss', 'val_loss', 'acc', 'val_acc'];
const container = {name: 'Model Training', styles: {height: '1000px'}};
const fitCallbacks = tfvis.show.fitCallbacks(container, metrics);
```



Sprite Sheet

- It's inefficient and a bad practice to load the data in one by one
- A way to solve the issue is by combining all the images to a sprite sheet and then combine it and then split it when importing it
- location of the MNIST sprite sheet: LINK

MNIST Classifier

- code:

```
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c9a1c230-7f19-452e-b566-72168fc3d0d0/MNIST.zip
```

Exercise

- Fashion MNIST Code:

```
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5b526978-5074-423a-947a-4377cb17e12d/FasionMNIST.zip
```