Live Visualisations of CNN's Activation Maps (Using Tensorflow.js)

Dip Patel Sep 7, 2018 · 3 min read

What are Activation Maps in CNN?

In simple terms, Activation Map is output of particular convolution layer. we can use activation maps for visualisation of CNN.
(here, Convolution layer is referred as Convolution layer followed by Max pooling.)

Are Activation maps helpful ?

Yes, They are useful for visualisation of CNN and we can also partially answer questions like : " How is CNN making decisions?", " Which things do CNN learn?"

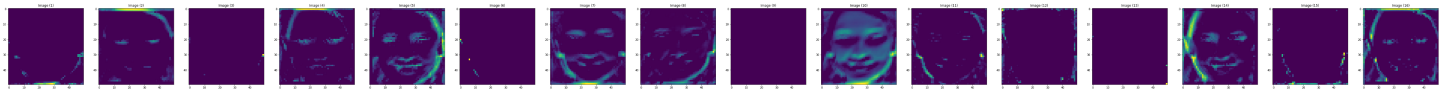
Remaining Blog Post is Divided into following Topics.

1. Detail of CNN
2. Converting Keras Model to Tensorflow.js Model
3. 3. Running Inference and Saving Activation Map
4. 4. Displaying Activation Map

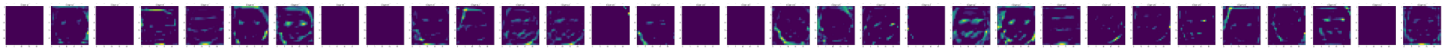
Details of CNN

For Live visualisation, We need smaller CNN which can output prediction in real-time even running on CPU. So, Any CNN with Fewer Convolution Layer will work. As Part of Machine Learning Course, I trained Compact CNN (3 Conv. Layer + 2 FC Layer) for Binary Gender Classification using Facial Image. CNN was trained on subset of IMDB-WIKI Face data-set[1] containing 40k Facial Images. Trained CNN has 0.94 F1-Score.

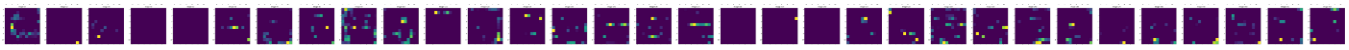
More Detail on CNN Architecture can be found here.



Activation Map of First Layer



Activation Map of Second Layer



Activation Map of Third Layer

Converting Keras Model to Tensorflow.js Model

CNN was implemented in Keras and saved as Keras Model. There is easy way of converting keras model to Tensorflow.js compatible Model.

For converting Model Following are steps. Official Documentation can be found here.

```
# install dependency
$ pip install tensorflowjs

# Converting Model
$ tensorflowjs_converter --input_format keras \
                         path/to/my_model.h5 \
                         path/to/tfjs_target_dir
```

Output Fill will be like:

```
$ ls tfjs_traget_dir
model.json
group1-shard1of1
```

```
group2-shard1of1
...
...
group6-shard1of1
```

Running Inference and Saving Activation Map

Now, It's time to predict gender using CNN model. For simplicity, We are assuming that we have facial Image coming from live feed.

You can check full code which contains implementation of extracting facial Image using tracking.js and then resizing into fix size of (50x50).

```
# loading Model
genderAI = await tf.loadModel('https://immortal3.github.io
/OnlineActivationMap/res/model/model.json');
```

We can direct use inference but then we will have only final result. We want to save activation Map during Forward pass for visualisation.

```
# Final Prediction
const genderprediction = model.predict(face_image);
```

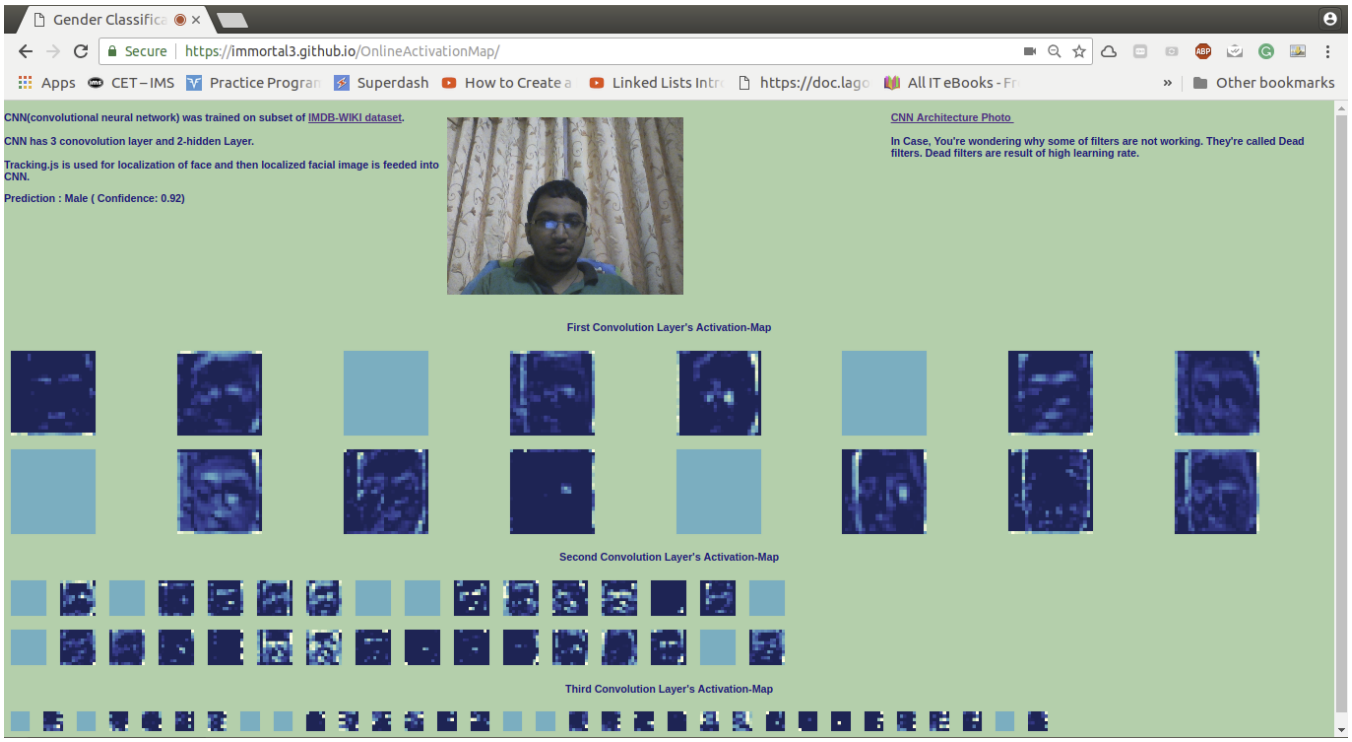
Saving Activation Map during Forward pass.

```
# list containing input for every layer
var input = []

# First Layer;s Input is Facial Image
input.push(tf.tidy(() => { return
tf.expandDims(facialImage,0).asType('float32').div(255.0)}));

# passing Input to every layer and saving output as input for next
layer
for (var i = 1; i <= 12; i++) {
```

```
    input.push(genderAI.layers[i].apply(input[i-1]));
}

# Saving activationmaps (note that we are displaying activation map
after applying max pool)
const firstconvactivationmap = input[2];
const secondconvactivationmap = input[4];
const thirdconvactivationmap = input[6];
```

Displaying Activation Map

For Visualisation, We can use heat-map from ploty.js. You can check out full code which includes implementation.



Live Demo : https://immortal3.github.io/OnlineActivationMap/

Full Code : https://github.com/immortal3/OnlineActivationMap/

About Me: https://immortal3.github.io/about/

(Note: For Mac Users, There're some problems with tensorflow.js. So, It may or may not work.)