

PROJECT REPORT

We created a web app using deep learning techniques which lets you visualize your convolutional neural network in a better way.

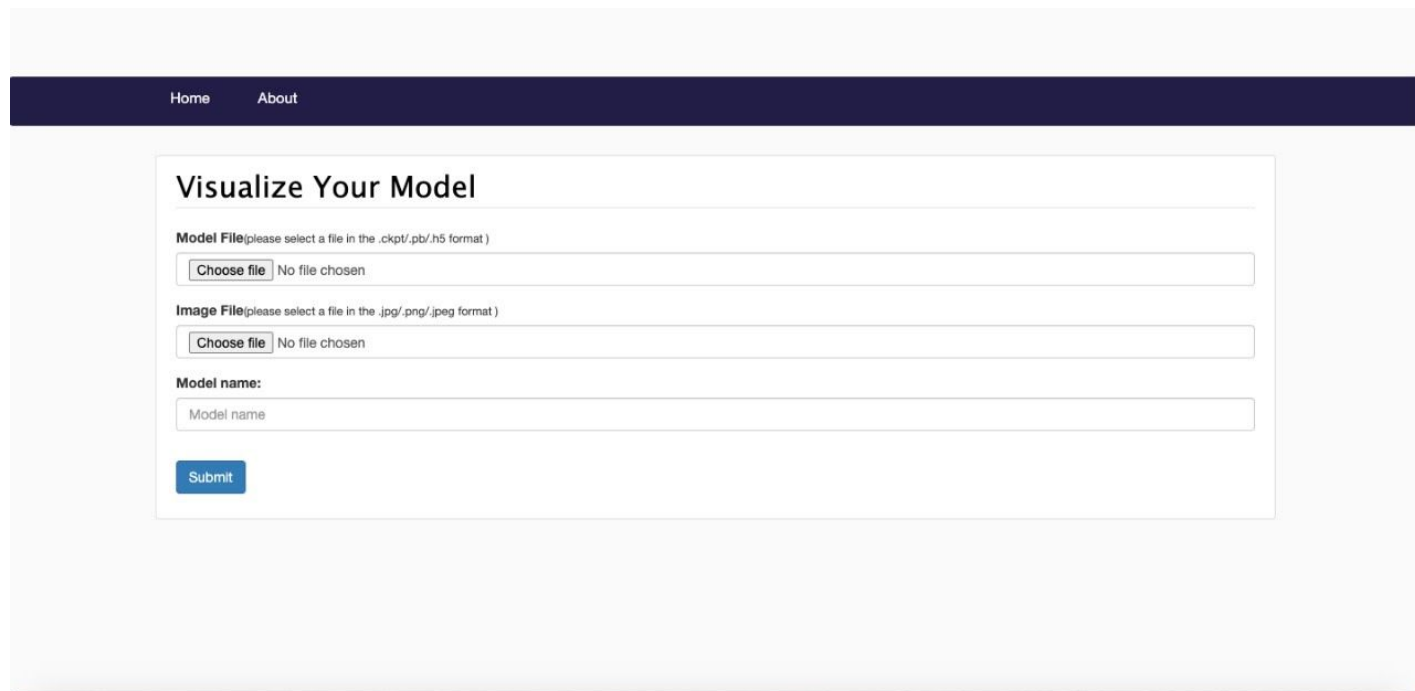
There are different ways to visualize model

1. Plotting model architecture.
2. Visualizing filters.
3. Activation maps.
4. Image occlusion.
5. Gradient .

The model which we have deployed uses activation maps. We used the Flask framework to deploy this model. Functionality of the model is described below -

Home Page :

Users are asked to upload a model file of learned weights of a neural network saved in .pb/ .ckpt or .h5 format. Any other formats are not allowed and will throw errors. Same way images are cross checked with the format allowed. All the files user uploaded are stored in the uploads folder using os.path.join. The web page for taking input looks like below.

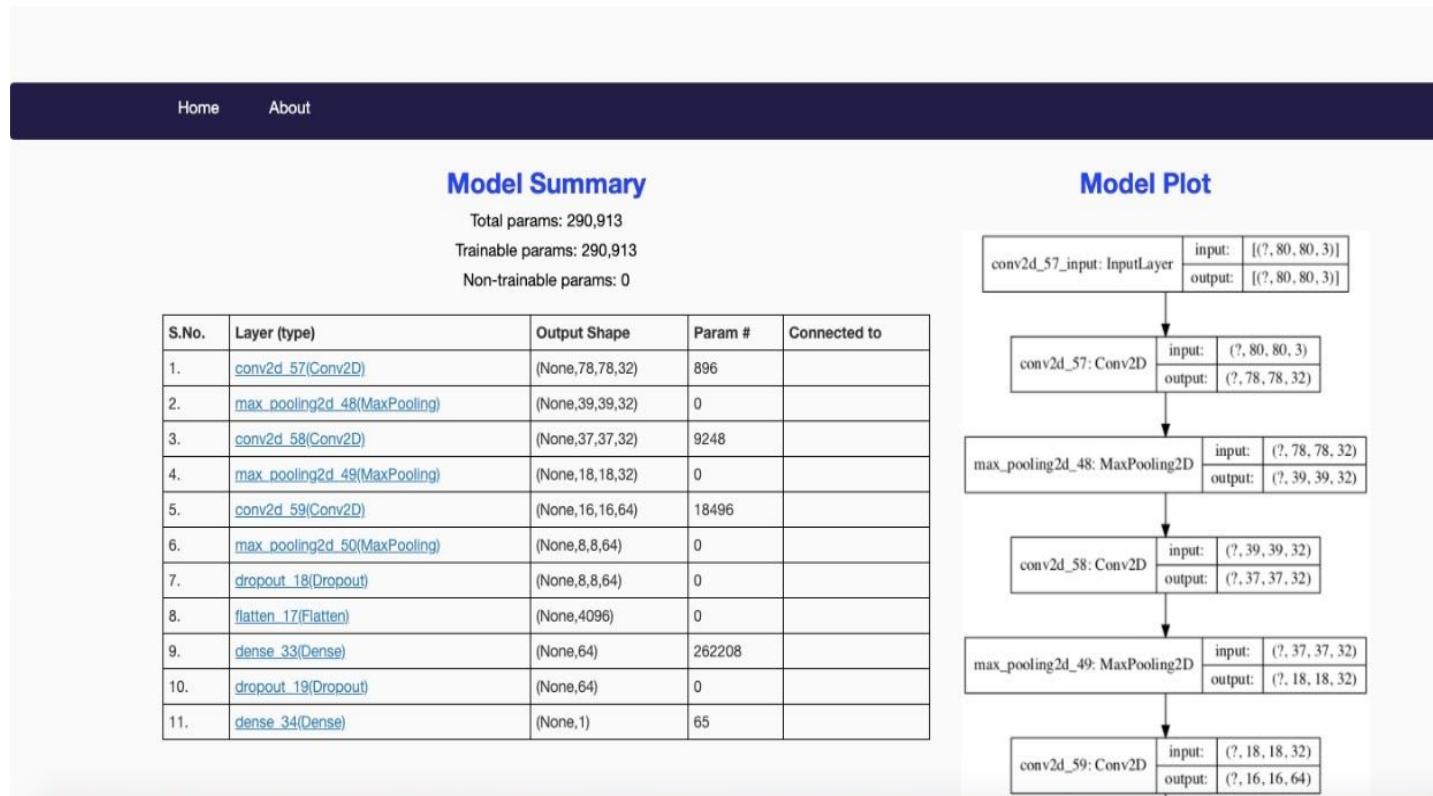


The screenshot displays a web application interface titled "Visualize Your Model". At the top, there is a dark blue navigation bar with the links "Home" and "About". The main content area is white and contains the following elements:

- Model File**(please select a file in the .ckpt/.pb/.h5 format)
A file selection input field with a "Choose file" button and the text "No file chosen".
- Image File**(please select a file in the .jpg/.png/.jpeg format)
A file selection input field with a "Choose file" button and the text "No file chosen".
- Model name:**
A text input field with the placeholder text "Model name".
- A blue "Submit" button.

Processing :

When the user clicks the submit button the model, which the user has uploaded ,is loaded using `tensorflow.keras.load()`.Then we display the model summary in which layers are clickable. This page also shows a Model plot, which is a flow chart of the neural network. The page looks as below



Final Output :

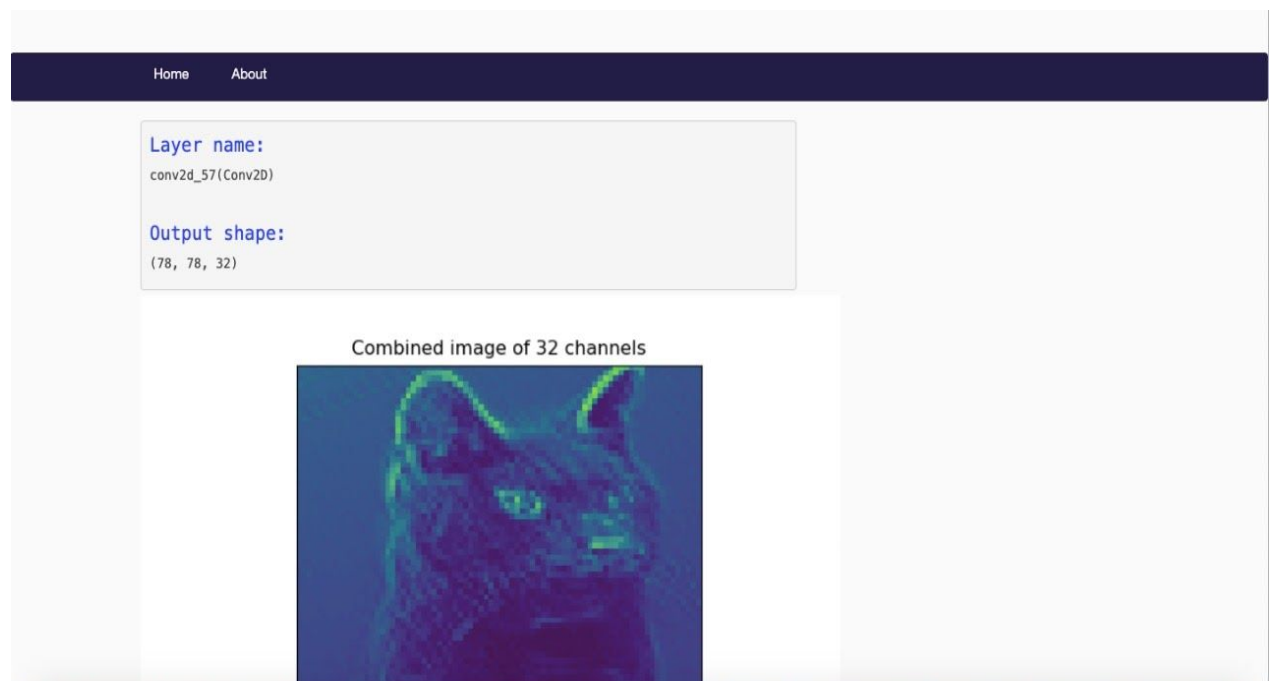
In the previous page each layer is clickable and when the user clicks on the layer to visualize we receive an id and name of the layer. We then convert the uploaded image into the allowed input size for the neural network. After that we pass the image . The model is trained in such a way that it gives output for the selected layer, this is achieved using

```
new_model = Model(input = model.inputs, outputs=model.layers[id].output)
```

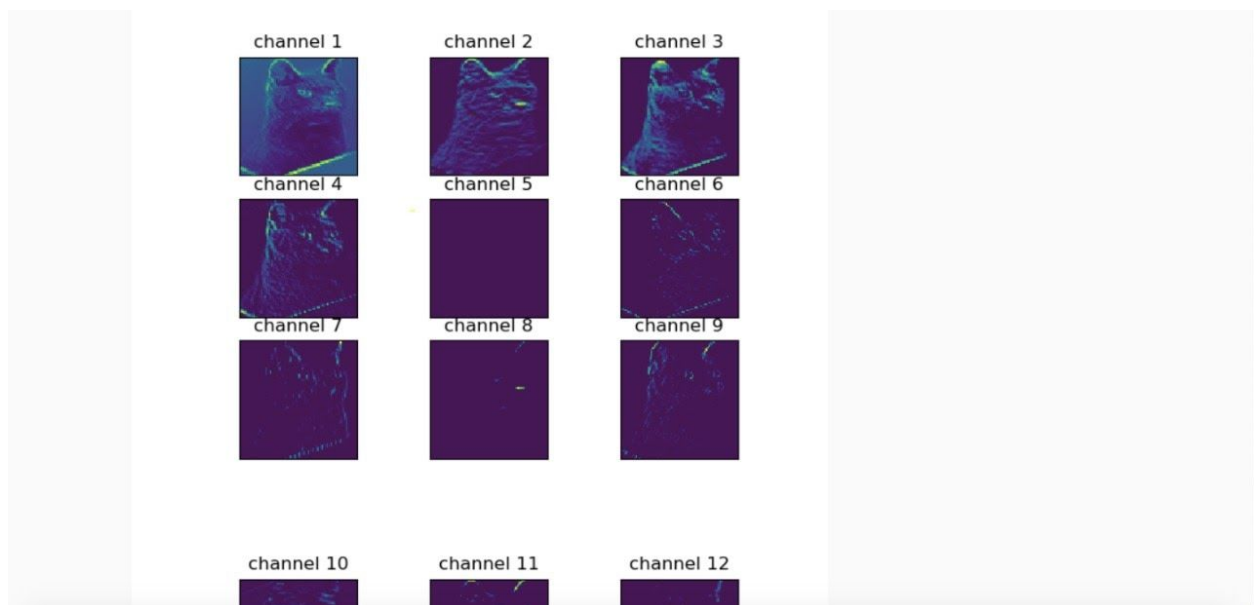
Then we pass the image to this new model and store the feature maps and display them.

```
Feature_maps = new_model.predict(img)
```

On this page we first display the concatenated image of all the channels, as below



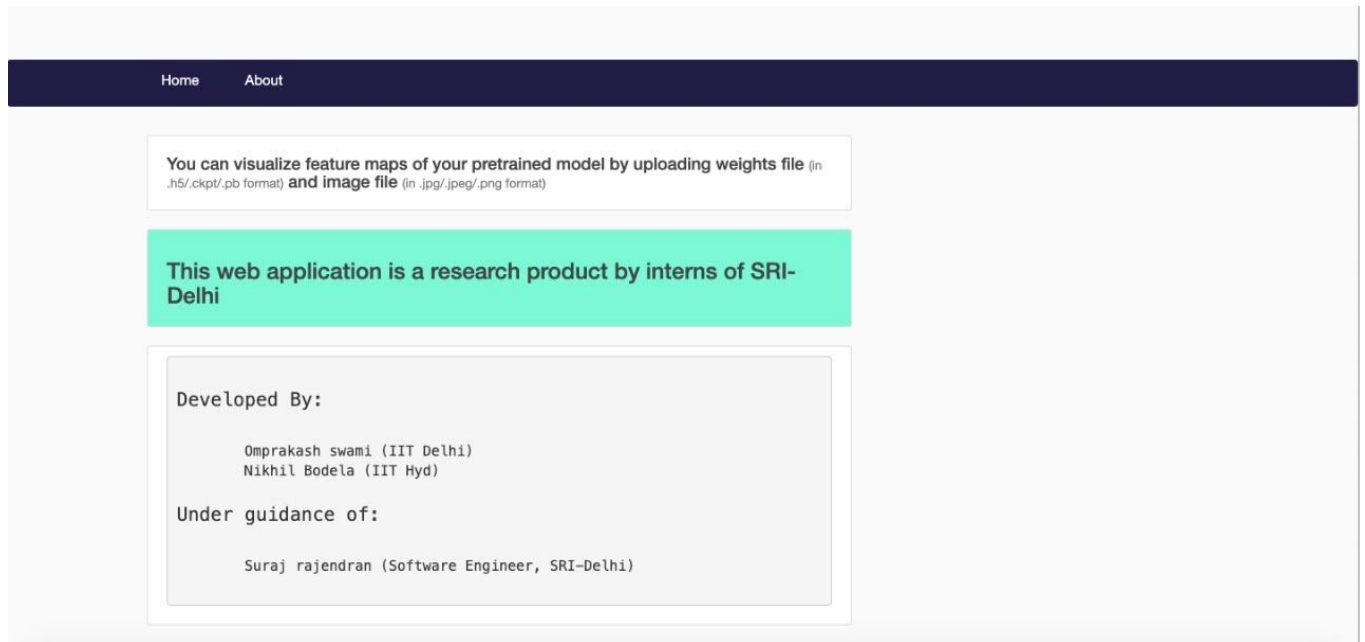
On this page , we also represent all the channel output as follows



About :

About page describes about the model and the method to use it -

About page looks as below -



All the above screenshot are for the dog cat classification model and the input image is



[Video link of how this app works.](#)

References :

- <https://www.analyticsvidhya.com/blog/2018/03/essentials-of-deep-learning-visualizing-convolutional-neural-networks/>
- <https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/>
- <https://arxiv.org/pdf/1311.2901.pdf>