**POC: Feature Importance calculation using SHAP visualizations**

**Pranav Kamesh S**

**Aim:** To find the feature importance for various features, i.e, quantify each feature’s effective contribution to changes in output wrt changes in input, using Shapley values and DeepExplainer objects, in Tensorflow/Keras framework, with low latency.

**Dataset**: MNIST - digits 0 to 9 recognition.

**How it works:** Shapley values, or SHAP values, are a measure of feature importance that uses an approximation of the DeepLift for evaluating the importance. These values attribute to each feature, the change in the expected model prediction, while conditioning on this feature. More details can be found [here](https://arxiv.org/pdf/1705.07874.pdf).

**Method:**

1. We train 2 models, one for 2 epochs and another for 10 epochs.
2. We make use of the SHAP package’s DeepExplainer, to calculate the SHAP values for both models, feeding the model and a background to the DeepExplainer object.
3. Visualize Image plots of the SHAP values to show the feature importance variation for different target images trained on the same background.

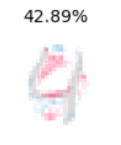
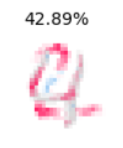
**Background used**: An image of the digit “0” is used as the background to create the DeepExplainer object. The model and background is fed to create the object, and we visualize the targets below:

**Interpretation:** Red pixels represent positive SHAP values that contributed to classifying that image as that particular class.

Blue pixels represent negative SHAP values that contributed to not classifying that image as that particular class.

The first image in each row represents the test image, with the following 10 images representing the explanation for why the model may think the test image can be each of the digits 0-9 respectively.

**Basic Examples:**

**  **

The first image represents a standard MNIST Target image. The 2nd and 3rd images show a Deep Explainer visualization of the number 4, after feeding the model and using 0 and 2 backgrounds, along with the probability of prediction of the 4. An explainer object tries to explain the model prediction basing it on the background image being used. This is why we see red and blue pixelation following the shape of the 0 and the 2, trying to explain which pixels cause such a prediction. This is also why we can see the clear images of 0 and 2 in the background in images 2 and 3. We also notice that the probability predictions are the same, which makes sense as the model predictions dont change with any change in the background for the explainer object.

**More Examples:**



Figure 1: Model trained for 10 epochs

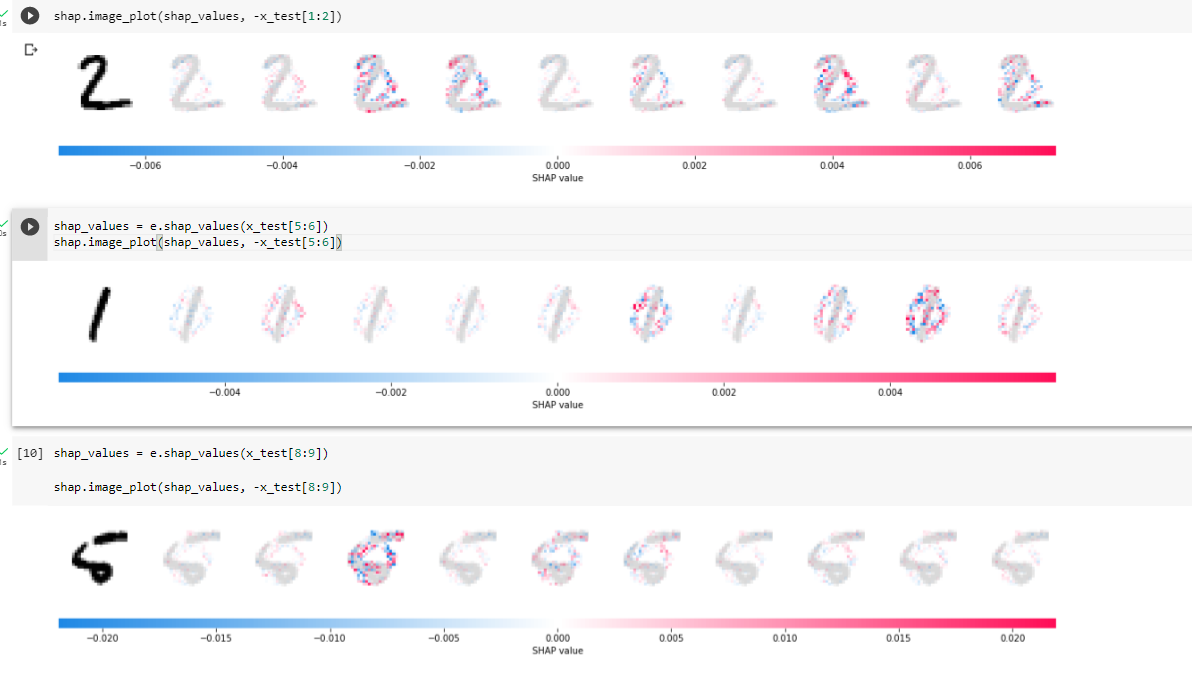


Figure 2: Model trained for 2 epochs

We can see that the explainer object shows a zero outline in most of the individual images. For the first 2 test images, we clearly see the high amounts of red in the 3rd and 2nd figure in that row, showing us why the model predicts those classes. High amounts of blue indicate that those pixels make the model strongly think that it is not an image of that class. High amounts of red indicate that those pixels make the model strongly think that it is an image of that class. We also can note that this DeepExplainer object is fed using a 0 background, but can be modified to use a background consisting of all the digits in the dataset.

Also, the increase in performance is clearly visible for the model trained on 10 epochs, when we compare the predicted probabilities below:



Figure 3: Model trained on 2 epochs: Predictions and Class probabilities



Figure 4: Model trained on 10 epochs: Predictions and Class probabilities

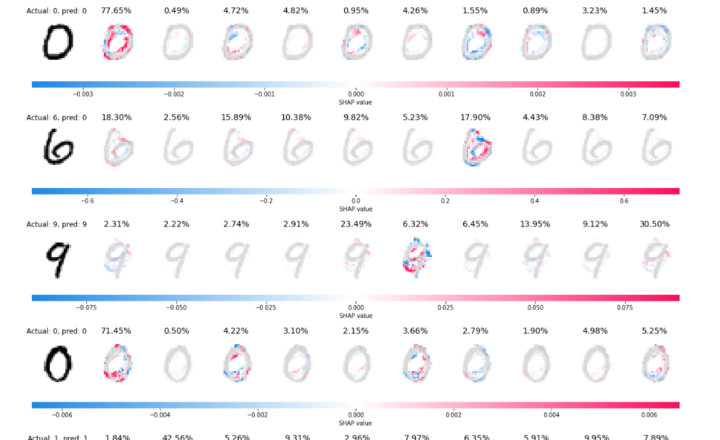


Figure 5: Visualizing Test Images

**Explanation**: For the 6 image, we see that 0 was predicted with a slightly higher probability. The high amount of blue pixelation in the 7th figure for 6, suggests that the model took those features to suggest that it was not a 6. Similarly for 9, despite some confusion with respect to predicting it as a 4, it eventually settles on 9.



Figure 6a and 6b: Visualizing more test images and explaining them

**Explanation:** In the first image, we see a 4 predicted instead of a 5 as the bottom portion curling upwards makes the model think it is a 4 instead. The “1” example in image 2 is interesting, as we can see that the red pixelation is in the centre for the target, as compared to the other images for the 1 which do not have red pixelation in the centre. This suggests that the model has learnt that the central vertical line is integral to predicting a one, and this is picked up by the explainer object.

**Method to obtain Feature Importance:**

For a standard classification problem, we can scale up the SHAP values to act as percentages, and multiply them with Δy, the change in output, to obtain the feature importance for each feature. This allows us to quantify the contribution of each feature towards change in input, which will be useful for our application.

**Application to our problem:**



**Figure 7:** Fair Market Value Model’s interface, with Feature importance

We first normalize/denormalize the inputs and outputs as required in keras\_trace\_model, and then port the Shapley code for creating DeepExplainer objects, into the models class. This will allow us to create DeepExplainer objects for predicting which features affect the output change the most (for the buy, sell and dealer prices), which can be visualized as shown in the image above.

The steps required would be:

1. Scale the Shapley values to obtain percentages.
2. Multiply with change in output quantity (y2 - y1) to obtain Feature Importance values.
3. Allow for option to choose initial point, so as to have Feature importance calculable for various timeframes.

**References:**

1. Slundberg and Lee -[A Unified Approach to Interpreting Model Predictions](https://arxiv.org/pdf/1705.07874.pdf)