SML Project

Description of project

Amir Pourmand [*1], Hossein Khalili [†2] and MohammadMahdi Abdollahpour [‡3]

[1,2,3]*MSc. AI at Sharif University of Technology*

Contents

Textolivre
Linguagem e Tecnologia

DOI: 000.yyyy.nnnnn

Corresponding author:
Amir Pourmand

Edited by:
Hamid R Rabiee

Received on:
October 22, 2020
Accepted on:
September 3, 2020
Published on:
July 16, 2021

*Email: pourmand1376@gmail.com
†Email: hosseinhkh@live.com
‡Email: mohammadmahdiabdollahpour@gmail.com

Abstract

This is our project report of what we've done and how we've done it. It mainly consists of explaining the interpretability methods and their use in our project. Of course, Some visualization to get us a clue of what's going on.

All team members participated in each part of the project:

1. Adbollahpour: LIME, BNN
2. Khalili: LRP
3. Pourmand: Kernel Shap, Report (Latex)

If you have any questions or problems with running each implementations, you can contact us and we will respond immediately.

Keywords: Interpretability. LIME. LRP. Kernel SHAP.

1 Introduction

If a machine learning model performs well, why do not we just trust the model and ignore why it made a certain decision? "The problem is that a single metric, such as classification accuracy, is an incomplete description of most real-world tasks." (DOSHI-VELEZ; KIM, 2017b)

Interpretable and explainable ML techniques emerge from a need to design intelligible machine learning systems, i.e. ones that can be comprehended by a human mind, and to understand and explain predictions made by opaque models, such as deep neural networks or gradient boosting machines. The early research on interpretable machine learning dates back to the 1990s and often does not refer to terms like "interpretability" or "explainability", not to mention that many classical statistical models can be deemed interpretable.(MARCINKEVIČS; VOGT, 2020)

When it comes to predictive modeling, you have to make a trade-off: Do you just want to know what is predicted? For example, the probability that a customer will churn or how effective some drug will be for a patient. Or do you want to know why the prediction was made and possibly pay for the interpretability with a drop in predictive performance? In some cases, you do not care why a decision was made, it is enough to know that the predictive performance on a test dataset was good. But in other cases, knowing the 'why' can help you learn more about the problem, the data and the reason why a model might fail. Some models may not require explanations because they are used in a low-risk environment, meaning a mistake will not have serious consequences, (e.g. a movie recommender system) or the method has already been extensively studied and evaluated (e.g. optical character recognition). The need for interpretability arises from an incompleteness in problem formalization, which means that for certain problems or tasks it is not enough to get the prediction (the what). The model must also explain how it came to the prediction (the why), because a correct prediction only partially solves your original problem. (DOSHI-VELEZ; KIM, 2017a)

Here we implement and visualize three most important methods that are LIME [1] , LRP [2] and KernelShap [3] respectively.

1 Local Interpretable Model-Agnostic
2 Layer-wise Relevance Propagation
3 SHapley Additive exPlanations

2 Selected Methods for Interpretability

2.1 LIME

Local surrogate models are interpretable models that are used to explain individual predictions of black box machine learning models. Instead of training a global surrogate model, LIME focuses on training local surrogate models to explain individual predictions. in order to get explanations we first select instance of interest for which we want to have an explanation of its black box prediction. The we perturb out dataset and get the black box predictions for these new points. We weight the new samples according to their proximity to the instance of interest. After that we train a weighted, interpretable model on the dataset with the variations and explain the prediction by interpreting the local model (RIBEIRO; SINGH; GUESTRIN, 2016).

For images it would not make much sense to perturb individual pixels, since many more than one pixel contribute to one class. Randomly changing individual pixels would probably not change the predictions by much. So, superpixels are usually used. We run a segmentation algorithm to obtain superpixels. Specifically, we use quick shift segmentation algorithm from Scikit package.

MNIST model achieves 57% accuracy after keeping the most important pixels. (evaluated on 2000 images of the test set)

2.2 LRP

What is LRP? lets take look at LRP OR Layer-wise relevance propagation method: Intuitively, what LRP does, is that it uses the network weights and the neural activations created by the forward-pass to propagate the output back through the network up until the input layer. There, we can visualize which pixels really contributed to the output. We call the magnitude of the contribution of each pixel or intermediate neuron "relevance" values R.

LRP is a conservative technique, meaning the magnitude of any output y is conserved through the backpropagation process and is equal to the sum of the relevance map R of the input layer. This property holds for any consecutive layers j and k, and by transitivity for the input and output layer.

Link

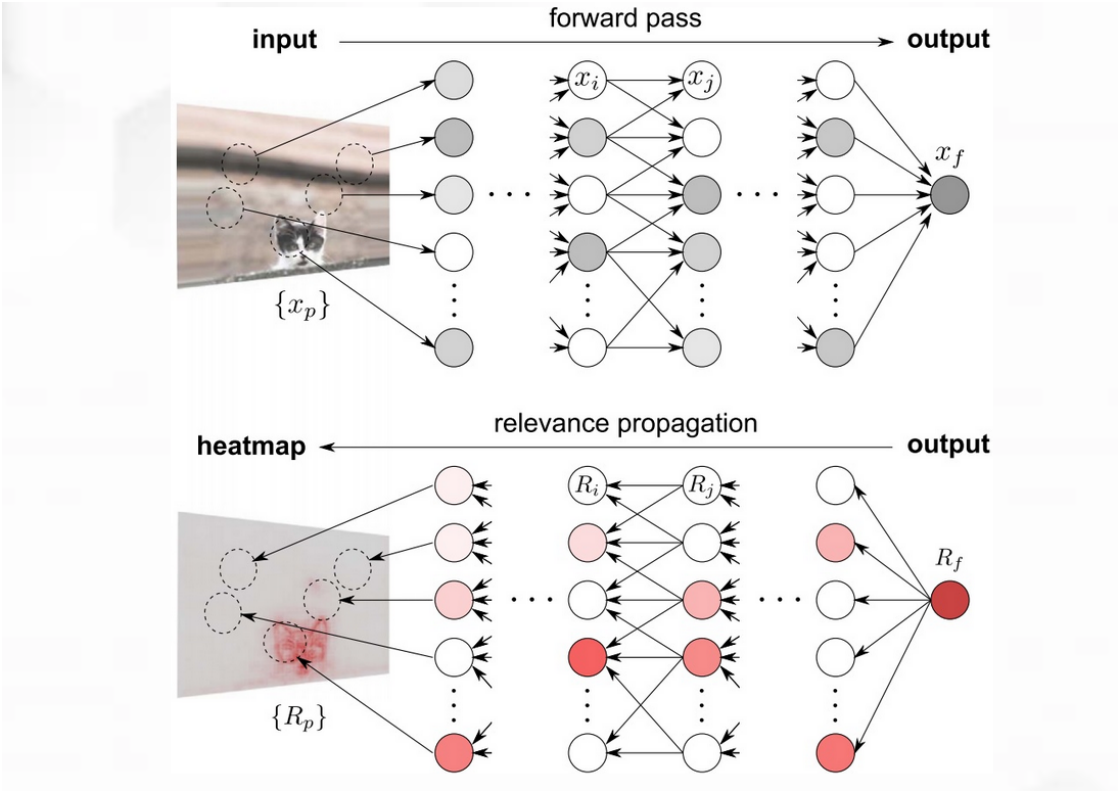


Figure 1. Generated Image from our network and its explanations

2.3 Kernel Shap (Extra)

SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions. (LUNDBERG; ERION, et al., 2020)

To understand how a single feature effects the output of the model we can plot the SHAP value of that feature vs. the value of the feature for all the examples in a dataset. Since SHAP values represent a feature's responsibility for a change in the model output, the plot below represents the change in predicted house price as RM (the average number of rooms per house in an area) changes. Vertical dispersion at a single value of RM represents interaction effects with other features. To help reveal these interactions we can color by another feature. If we pass the whole explanation tensor to the color argument the scatter plot will pick the best feature to color by. In this case it picks RAD (index of accessibility to radial highways) since that highlights that the average number of rooms per house has less impact on home price for areas with a high RAD value (LUNDBERG; NAIR, et al., 2018) (LUNDBERG; LEE, 2017)

3 Our Model (Network)

MNIST model consists of two convolutional layers with kernels size of five followed by two fully connected layers. CIFAR10 model consists of three convolutional blocks and three fully connected layers with relu activation function. Each convolutional block consists of first a convolutional layer with kernel size of 3 and 32 channels then a batch normalization layer and a relu activation function. Then it has a convolutional layer with kernel size of 3 and 64 channels and a relu activation function followed by max pooling.

3.1 MNIST

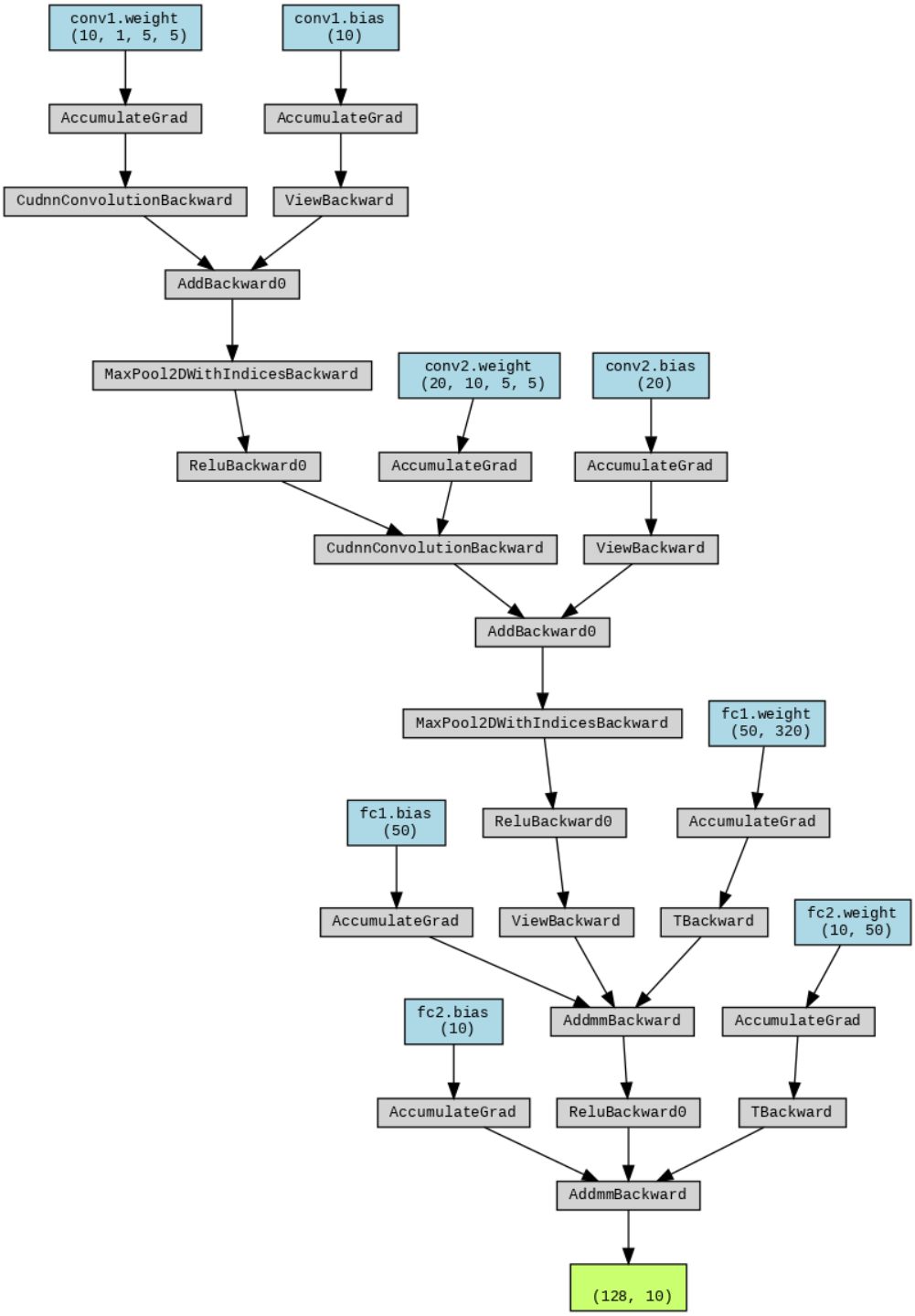


Figure 2. Architecture of MNIST model

3.2 CIFAR10

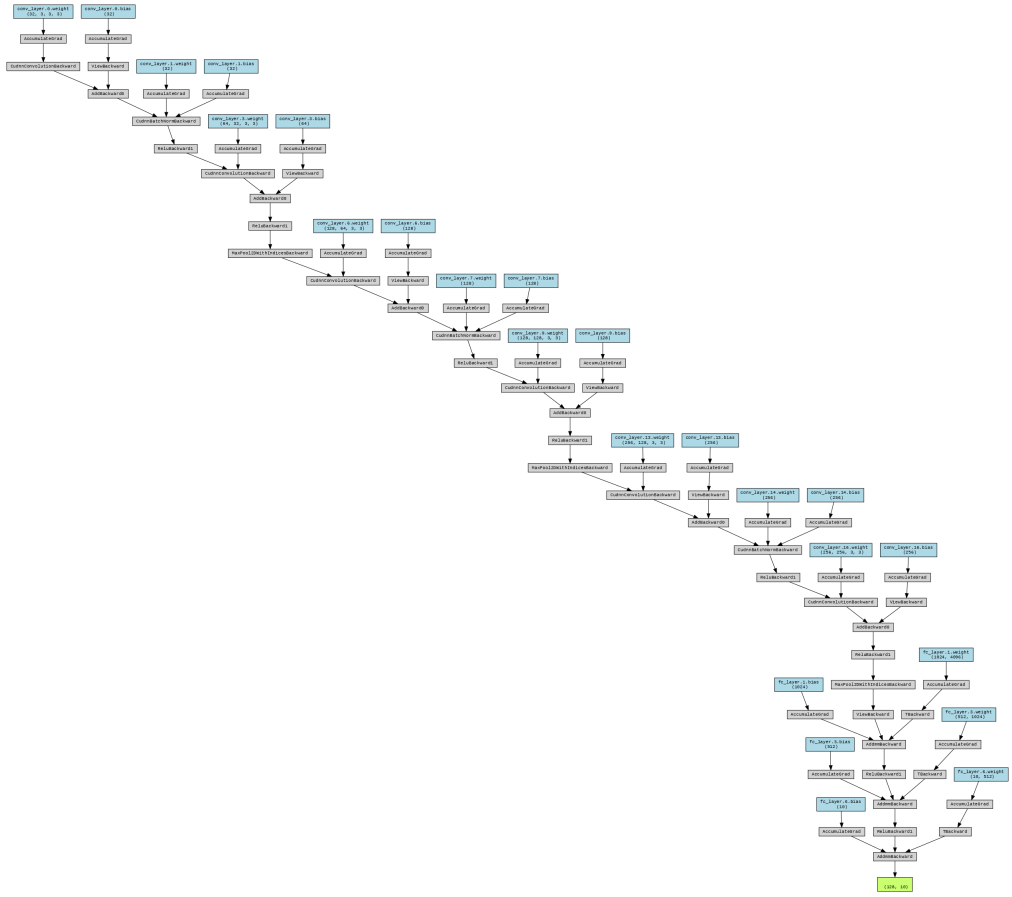


Figure 3. Architecture of CIFAR10 model

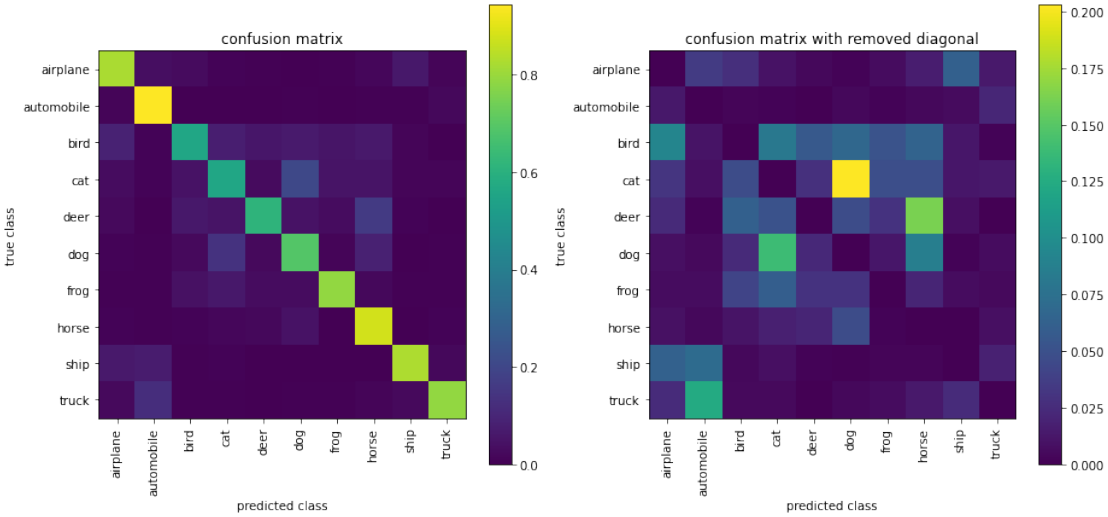


Figure 4. Confusion Matrix of our Method

4 Result on Test Data

MNIST model achieves 98% accuracy on the test set and CIFAR10 model achieves 91% accuracy on the test set.

5 Implementation

Here we take a look at our most important functions to implement each method.

5.1 LIME

There are most important libraries used for LIME:

```
import numpy as np
import scipy as sp
from sklearn.linear_model import Ridge, lars_path
from sklearn.utils import check_random_state
from skimage.segmentation import felzenszwalb, slic, quickshift
import copy
from functools import partial
import numpy as np
import sklearn
from skimage.color import gray2rgb
```

explain instance with data produces score for each feature (segment):

```
class LimeBase(object):

    def forward_selection(self, data, labels, weights, num_features):

    def feature_selection(self, data, labels, weights, num_features, method):

    def explain_instance_with_data(self, neighborhood_data, neighborhood_labels,
                                                distances, label, num_features,
                                                feature_selection='auto',
                                                model_regressor=None):
```

Explainer produces explanations for an specific instance, explain instance creates neighboring data and calls explain instance with data

```
class LimeImageExplainer(object):

    def explain_instance(self, image, classifier_fn, labels=(1,), hide_color=None,
                                                top_labels=5, num_features=100000,
                                                num_samples=1000,  batch_size=10,
                                                distance_metric='cosine'):
```

Explanation produced by Explainer can produce the masks for us. Masks are applied to the source image and then visualized in visualization section.

```
class ImageExplanation(object):
    def get_image_and_mask(self, label, positive_only=True, negative_only=False,
                                                hide_rest=False, num_features=5,
                                                min_weight=0.):
```

5.2 LRP

we should create conv layer and linear layer and max pool

```
class Conv2d(torch.nn.Conv2d):
    def _conv_forward_explain(self, input, weight, conv2d_fn, **kwargs):
        if self.padding_mode != 'zeros':
            return conv2d_fn(F.pad(input, self._reversed_padding_repeated_twice,
                                                    mode=self.padding_mode),
                             weight, self.bias, self.stride,
                             _pair(0), self.dilation, self.groups, **kwargs)

        p = kwargs.get('pattern')
        if p is not None:
            return conv2d_fn(input, weight, self.bias, self.stride, self.padding,
                                                    self.dilation, self.groups,
                                                    p)
        else: return conv2d_fn(input, weight, self.bias, self.stride, self.padding,
                                                self.dilation, self.groups)


    def forward(self, input, explain=False, rule="epsilon", **kwargs):
        if not explain: return super(Conv2d, self).forward(input)
        return self._conv_forward_explain(input, self.weight, conv2d[rule], **kwargs
                                                )
```

```
class Linear(torch.nn.Linear):
    def forward(self, input, explain=False, rule="epsilon", **kwargs):
        if not explain: return super(Linear, self).forward(input)

        p = kwargs.get('pattern')
        if p is not None: return linear[rule](input, self.weight, self.bias, p)
        else: return linear[rule](input, self.weight, self.bias)
```

```
class MaxPool2d(torch.nn.MaxPool2d):
    def forward(self, input, explain=False, rule="epsilon", **kwargs):
        if not explain: return super(MaxPool2d, self).forward(input)
        return maxpool2d[rule](input, self.kernel_size, self.stride, self.padding)
```

Also LRP has some rules and we should select best rule for our project we test and use alpha1beta0 rule like table 14. (MONTAVON et al., 2019)

After that we should convert our pretrained model to lrp model so we need a convertor like this:

```
def convert_vgg(module, modules=None):
    # First time
    if modules is None:
        modules = []
        for m in module.children():
            convert_vgg(m, modules=modules)


            if isinstance(m, torch.nn.AdaptiveAvgPool2d):
                modules.append(torch.nn.Flatten())

        sequential = Sequential(*modules)
        return sequential

    # Recursion
    if isinstance(module, torch.nn.Sequential):
        for m in module.children():
            convert_vgg(m, modules=modules)

    elif isinstance(module, torch.nn.Linear) or isinstance(module, torch.nn.Conv2d):
```

Name	Formula	Usage	DTD
LRP-0 [7]	$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k$	Upper layers	✓
LRP-ϵ [7]	$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_{0,j} a_j w_{jk}} R_k$	Middle layers	✓
LRP-γ	$R_j = \sum_k \frac{a_j (w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j (w_{jk} + \gamma w_{jk}^+)} R_k$	Lower layers	✓
LRP-$\alpha\beta$ [7]	$R_j = \sum_k \left(\alpha \frac{(a_j w_{jk})^+}{\sum_{0,j} (a_j w_{jk})^+} - \beta \frac{(a_j w_{jk})^-}{\sum_{0,j} (a_j w_{jk})^-} \right) R_k$	Lower layers	$\times^a$
flat [30]	$R_j = \sum_k \frac{1}{\sum_j 1} R_k$	Lower layers	×
w^2-rule [36]	$R_i = \sum_j \frac{w_{ij}^2}{\sum_i w_{ij}^2} R_j$	First layer ($\mathbb{R}^d$)	✓
$z^{\mathcal{B}}$-rule [36]	$R_i = \sum_j \frac{x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_i x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-} R_j$	First layer (pixels)	✓

(^{a}DTD interpretation only for the case $\alpha = 1, \beta = 0$.)

Figure 5. non-exhaustive list of propagation rules that are commonly used for explaining deep neural networks with ReLU nonlinearities

```
        class_name = module.__class__.__name__
        lrp_module = conversion_table[class_name].from_torch(module)
        modules.append(lrp_module)

    elif isinstance(module, torch.nn.ReLU):

        modules.append(torch.nn.ReLU())
    else:
        modules.append(module)
```

then we easily run our code to genrate images and calculate accuracy

```
vgg = vgg16()
checkpoint = torch.load('./models/cifar10_vgg16.tar')
vgg.load_state_dict(checkpoint['state_dict'])
vgg.to(device)
vgg.eval()
lrp_vgg = convert_vgg(vgg).to(device)
```

backward and calculate the explanation heat map

```
y_hat = model.forward(X, explain=True, rule=rule)
y_hat = y_hat[torch.arange(X.shape[0]), y_hat.max(1)[1]]
y_hat = y_hat.sum()
y_hat.backward()
attr = X.grad
attr = heatmap_grid(attr, cmap_name='seismic')
```

for see our result you can just run 'please_run_cifar10_vgg16.py'or 'please_run_minist.py'

5.3 Kernel Shap

Needed libraries are as follows:

```
import pickle
import inspect
import logging
import cloudpickle
import pandas as pd
import numpy as np
import scipy as sp
import sys
import warnings
import copy
import operator
import sklearn
```

This is the main class which we used to implement our Shap. Here is its functions.

```
class Deep(Explainer):
    def __init__(self, model, data, session=None, learning_phase_flags=None):
    def shap_values(self, X, ranked_outputs=None, output_rank_order='max',
                                                  check_additivity=True):
```

We also have TeacherForcing class which help us understand its relation to input.

```
class TeacherForcing(Model):
        def __init__(self, model, tokenizer=None, similarity_model=None,
                                                    similarity_tokenizer=None,
                                                    batch_size=128, device=None):
    def __call__(self, X, Y):
    def update_output_names(self, output):
    def get_output_names(self, output):
    def get_outputs(self, X):
    def get_inputs(self, X, padding_side='right'):
    def get_logodds(self, logits):
    def model_inference(self, inputs, output_ids):
    def get_teacher_forced_logits(self, X, Y):
    def save(self, out_file):
    def load(cls, in_file, instantiate=True):
```

We also needed explainer that uses Serializer which we defined as below (note that these modules are not essential for the algorithm but essential to send data between classes and models):

```
import copy
import numpy as np
import scipy as sp

class Explainer(Serializable):
    def __init__(self, model, masker=None, link=None, algorithm="auto", output_names
                                                  =None, feature_names=None, **kwargs)
                                                  :
    def __call__(self, *args, max_evals="auto", main_effects=False, error_bounds=
                                                  False, batch_size="auto", outputs=
                                                  None, silent=False, **kwargs):
    def supports_model_with_masker(model, masker):
    def _compute_main_effects(fm, expected_value, inds):
    def save(self, out_file, model_saver=".save", masker_saver=".save"):
```

and Serializer itself is defined as below:

```
class Serializer():
    def __init__(self, out_stream, block_name, version):
    def __enter__(self):
    def __exit__(self, exception_type, exception_value, traceback):
    def save(self, name, value, encoder="auto"):
```

6 Visualization

In this part, we will see some visualizations of each of our methods but all visualizations from our experiments are included in the appendix.

6.1 LIME

Here at, we can see examples of interpretability of LIME on both MNIST and CIFAR10 datasets.

6.1.1 MNIST

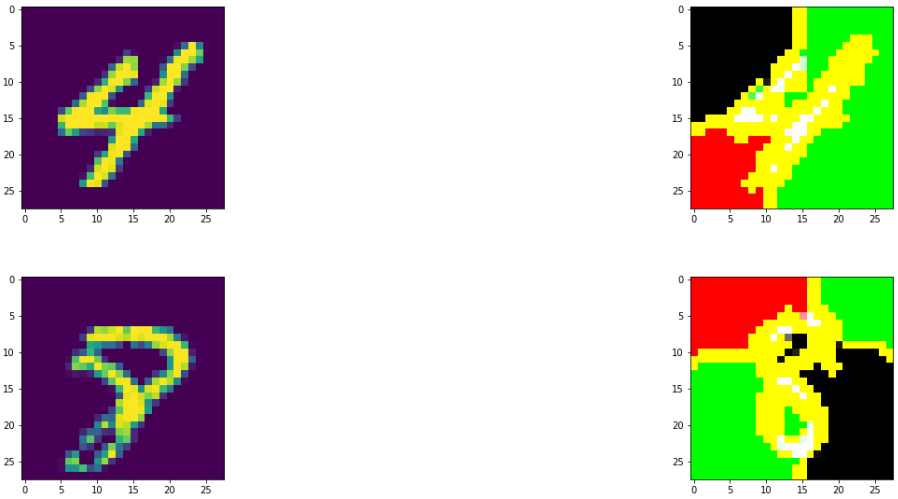


Figure 6. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

6.1.2 CIFAR10

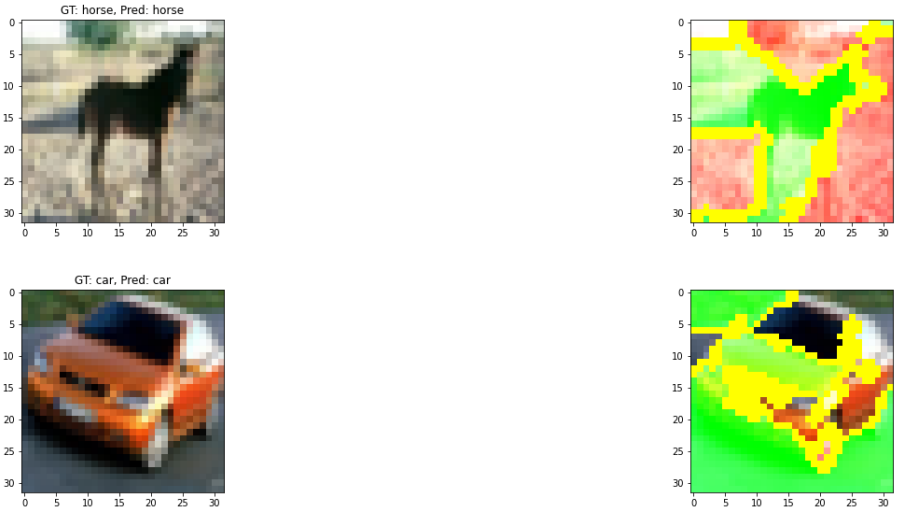


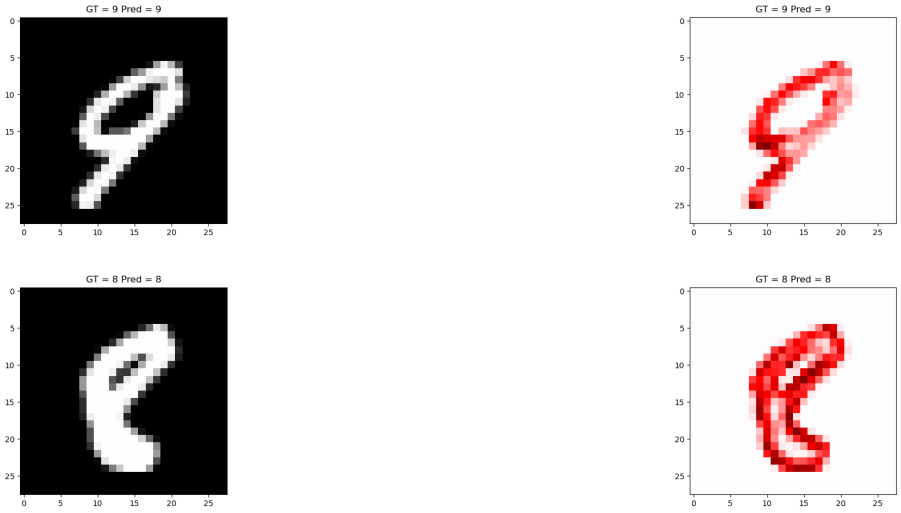
Figure 7. Original and Lime explanation of CIFAR10 images. Images in left column are original images and images in the right column are LIME explanations.

6.2 LRP

Here at, we can see examples of interpretability of LRP on both MNIST and CIFAR10 datasets.

6.2.1 MNIST

Figure 8. Original and LRP explanation of MNIST images. Images in left column are original images and images in the right column are LRP explanations.



6.2.2 CIFAR10

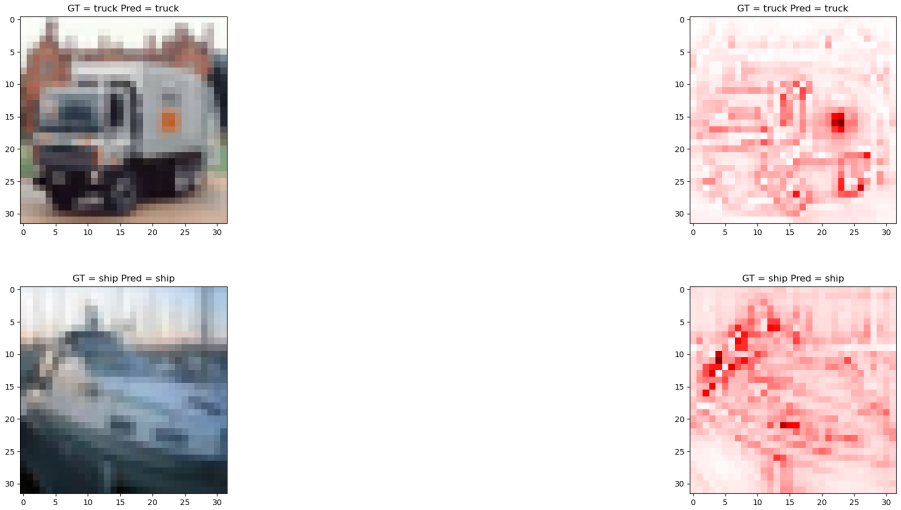


Figure 9. Original and LRP explanation of CIFAR10 images. Images in left column are original images and images in the right column are LRP explanations.

6.3 Kernel SHAP

Here at, we can see examples of interpretability of kernelshap on both MNIST and CIFAR10 datasets.

6.3.1 MNIST

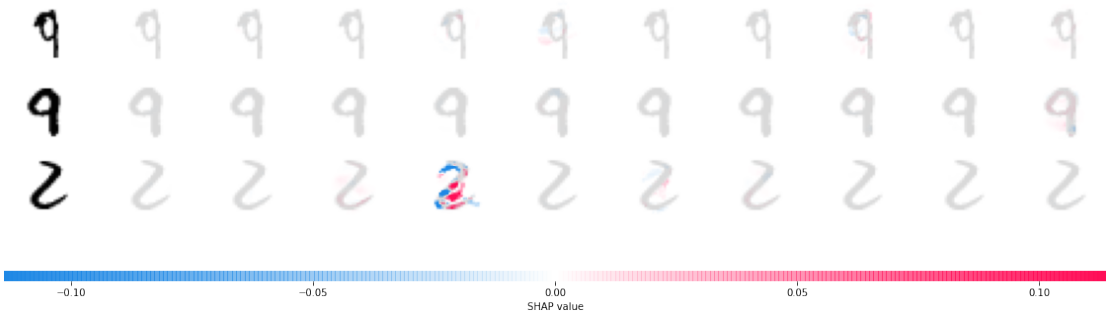


Figure 10. Kernel Shap Interpretation Sampled Image 9

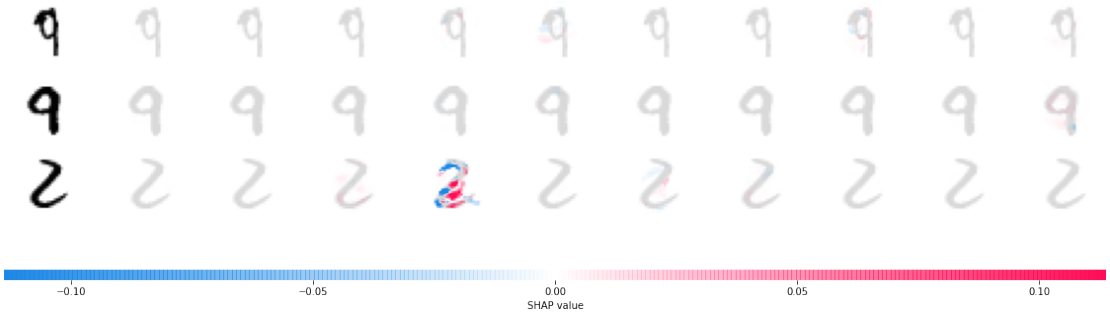


Figure 11. Kernel Shap Interpretation Sampled Image 10

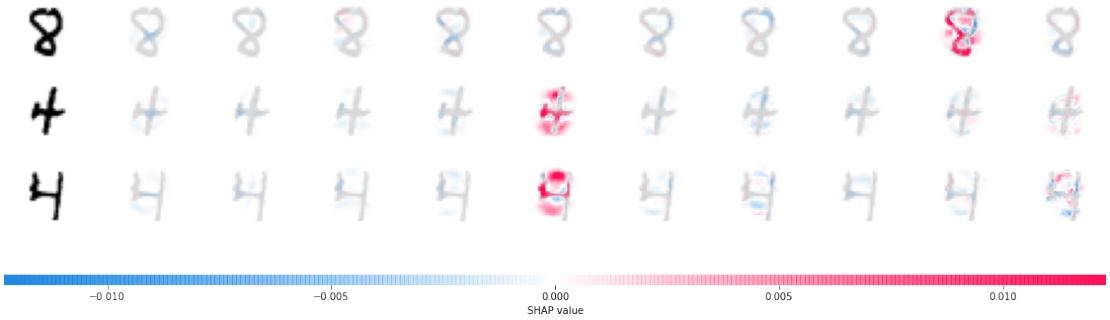


Figure 12. Kernel Shap Interpretation Sampled Image 11

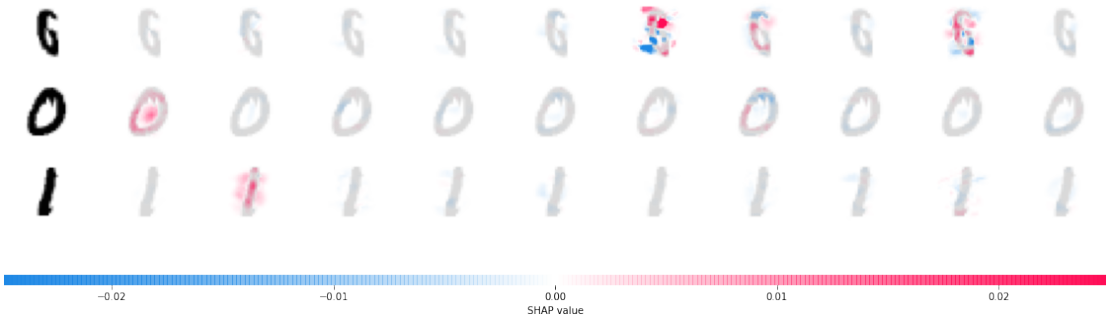


Figure 13. Kernel Shap Interpretation Sampled Image 3

6.3.2 CIFAR10

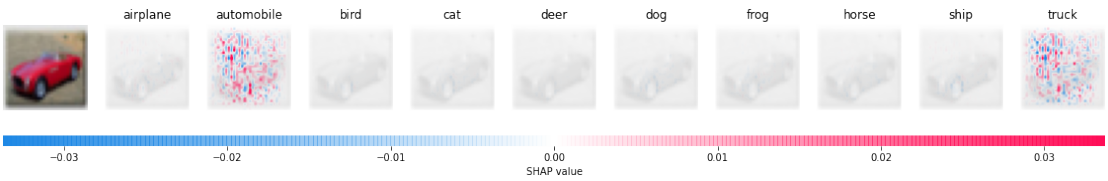


Figure 14. Generated Image from our network and its explanations

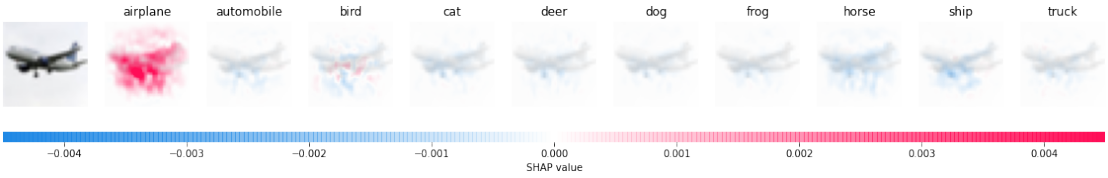


Figure 15. Kernel Shap Interpretation Sampled Image 1

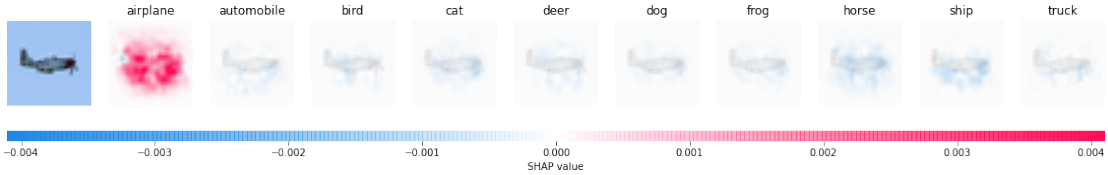


Figure 16. Kernel Shap Interpretation Sampled Image 2

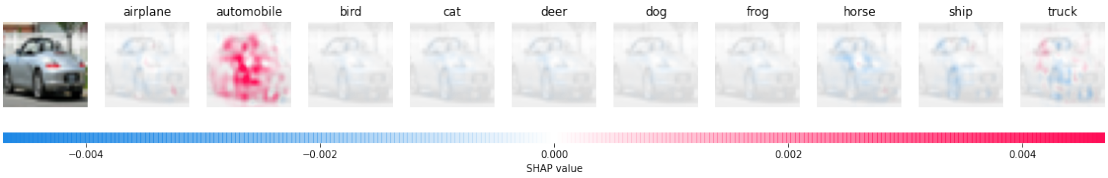


Figure 17. Kernel Shap Interpretation Sampled Image 3

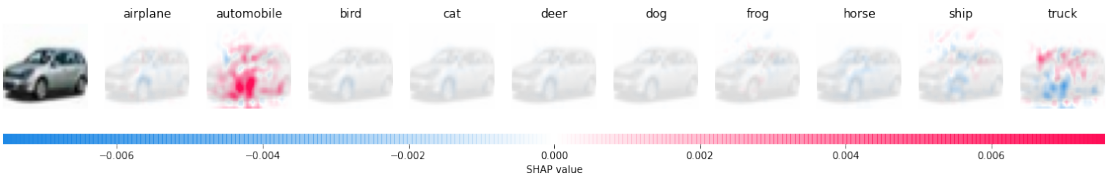


Figure 18. Kernel Shap Interpretation Sampled Image 4

7 Comparison of Pixel Effects

Here, we compare how random pixel effect in Phase 5 affects each method.

Table 1. Comparison of Pixel removal effect

Dataset / Method	LIME	LRP	Shap
MNIST Before	97	98.5	98
MNIST After	57	98.2	61
CIFAR10 Before	90	91.7	91
CIFAR 10 After	13	22.5	17

Here we can see some CIFAR images in which we change according to instructions given.

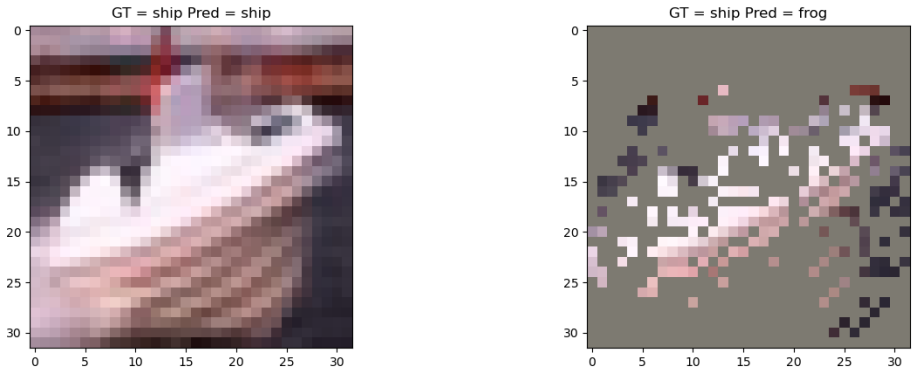


Figure 19. A Ship that only important pixels are left as it as

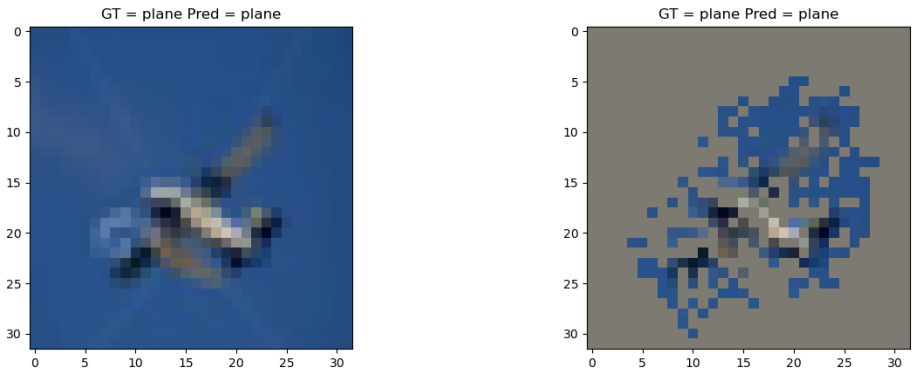


Figure 20. A Plane that only important pixels are left as it as

8 Model Optimization (Optional)

For optimization of model and accuracy, We have used our interpretability models for data augmentation in both datasets. In this technique, we keep important pixels (pixels that are considered important by our model for instance LRP) and we randomize other pixels or we substitute them with pixels in other images. This is called data augmentations (If you want to take a look at specific code, you can look at LRP folder)

By this techinque, we could increase the accuracy of the model on test data by 2 percent which is excellent.

Note: we have also uploaded the trained model file to Google Drive

9 Bayesian Interpretability

We used an open source implementation at GitHub of BNN for our experiments. Figure21 shows architecture of the BNN. We used stack of two bayesian fully connected layers and a relu actication function between them. This model achieved 96% accuracy on the test set. Table 2 shows effect of removing least contributing pixels (evaluated on 2000 images of test set). Accuracy of BNN on the chosen 2000 samples of test set is 95.05%. This results show that 10% of the most important pixels gives a good accuracy and small gains are obtained by adding more (less informative) pixels (in out case segments). Figure 21 shows a diagram of the model.

Table 2. Comparison of Pixel removal effect

Percentage of Keeping	10	20	30	40	50
Accuracy	54.1	54.5	55.2	62.35	67.2

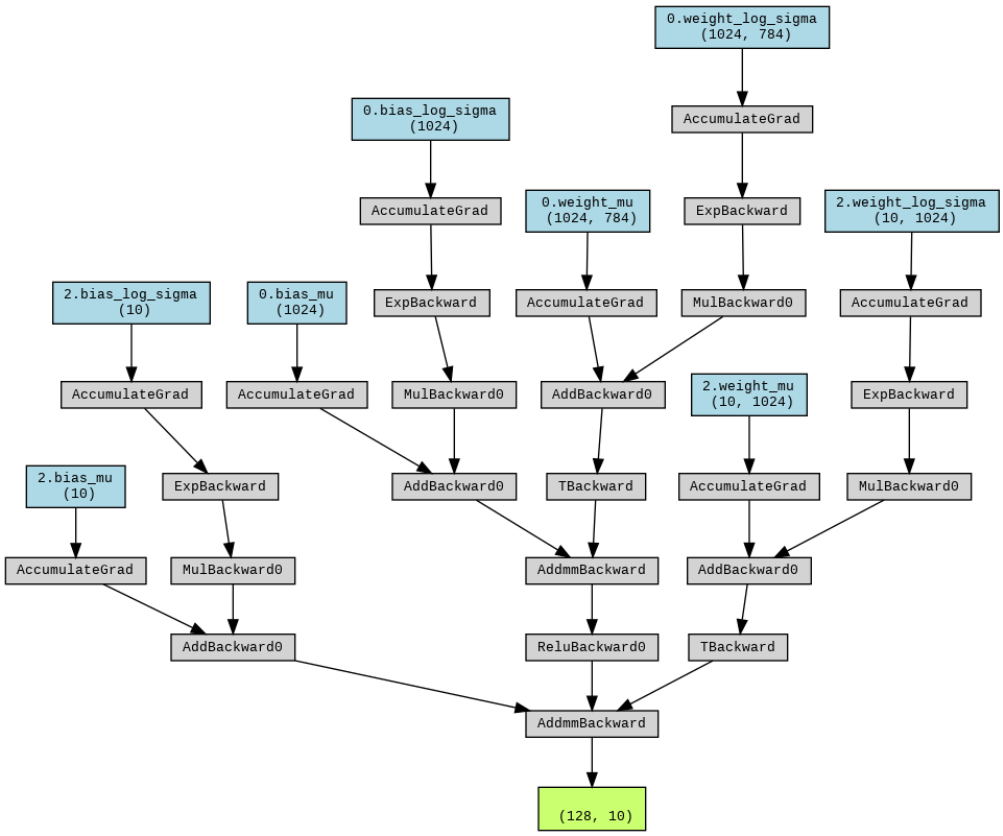


Figure 21. Architecture of BNN model

9.1 MNIST

Figure 22 shows some samples of original image and LIME explanation of the image by the BNN from MNIST dataset. Full set of exmaples are available in Appendix.

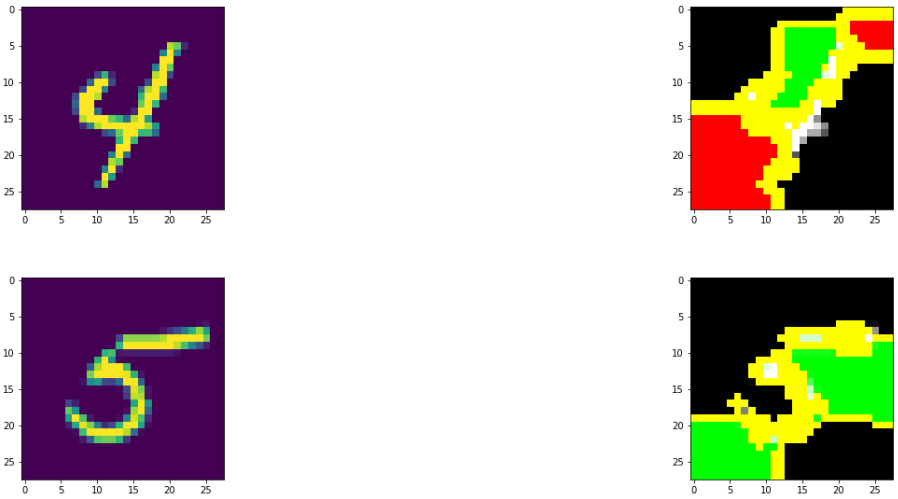


Figure 22. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

10 Summary

Here we trained models to classify MNIST and CIFAR10 and fortunately we got good test results on both datasets (MNIST: 98%, CIFAR10:91%). Then we implemented and also visualized LIME, LRP and SHAP and we saw that all of them can be used in real-world scenarios as they can somehow understand the images better!

After keeping important pixels and masking out less informative ones CIFAR10 accuracy drops more than MNIST because MNIST has a black background and masking many pixels does not make much difference. Also, in this experiment, LRP accuracy drops less since it produces better and fine grained (in LIME we score the segments) explanations. on MNIST, LRP gives high scores to pixels presenting the digit and and byt masking the other pixels, image does not significantly chnage so LRP accuracy drops less than 1%.

References

DOSHI-VELEZ, Finale; KIM, Been. *Towards A Rigorous Science of Interpretable Machine Learning*. [S.l.: s.n.], 2017. arXiv: 1702.08608 [stat.ML].

DOSHI-VELEZ, Finale; KIM, Been. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

LUNDBERG, Scott M; LEE, Su-In. A Unified Approach to Interpreting Model Predictions. In: GUYON, I. et al. (Eds.). *Advances in Neural Information Processing Systems 30*. [S.l.]: Curran Associates, Inc., 2017. p. 4765–4774. Available from: <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>.

LUNDBERG, Scott M; NAIR, Bala, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, Nature Publishing Group, v. 2, n. 10, p. 749, 2018.

LUNDBERG, Scott M.; ERION, Gabriel, et al. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, Nature Publishing Group, v. 2, n. 1, p. 2522–5839, 2020.

MARCINKEVIČS, Ričards; VOGT, Julia E. Interpretability and explainability: A machine learning zoo mini-tour. *arXiv preprint arXiv:2012.01805*, 2020.

MONTAVON, Grégoire et al. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, Springer, p. 193–209, 2019.

RIBEIRO, Marco Tulio; SINGH, Sameer; GUESTRIN, Carlos. " Why should i trust you?" Explaining the predictions of any classifier. In: PROCEEDINGS of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. [S.l.: s.n.], 2016. p. 1135–1144.

11 Appendix

11.1 MNIST LIME

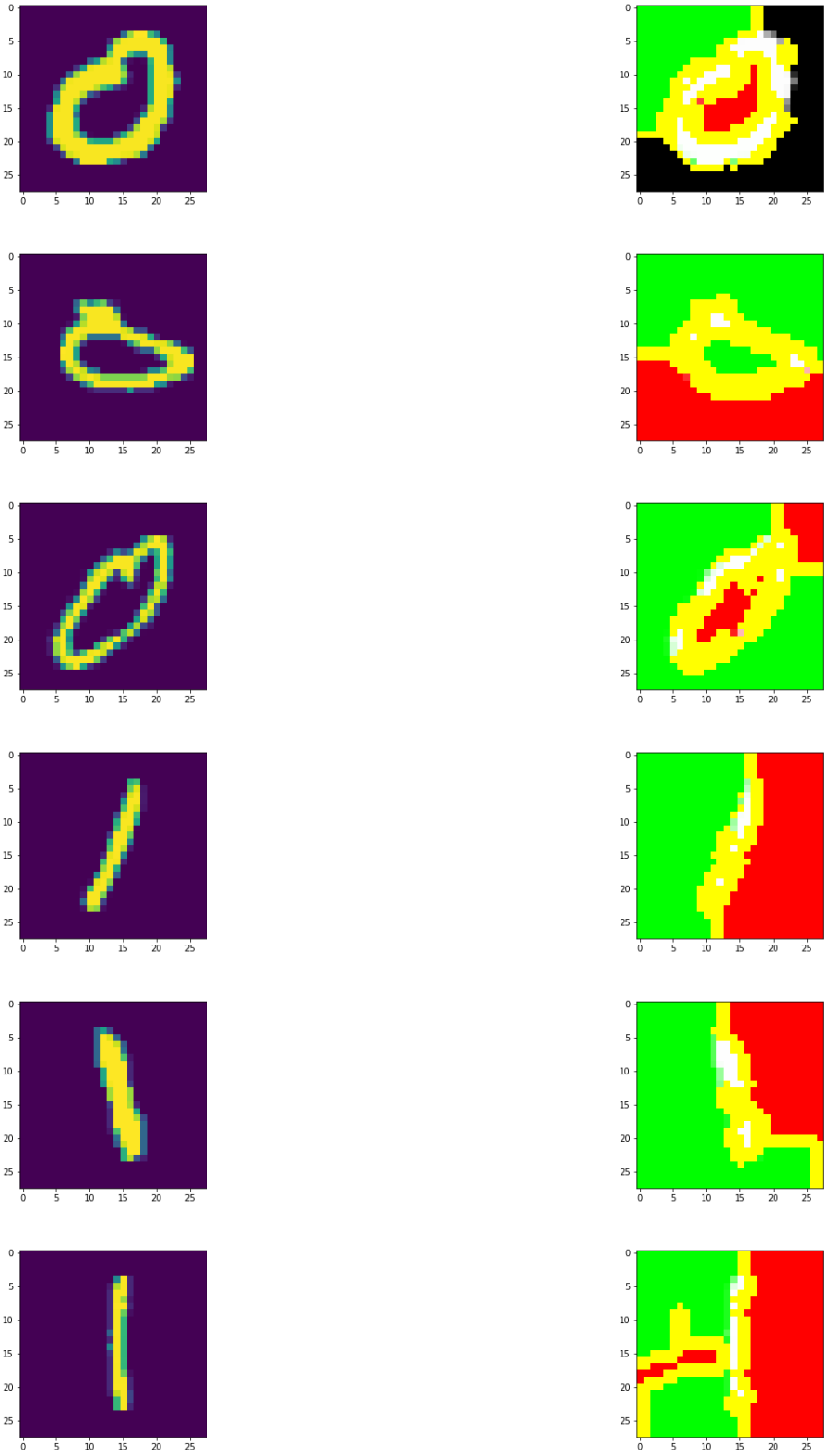


Figure 23. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

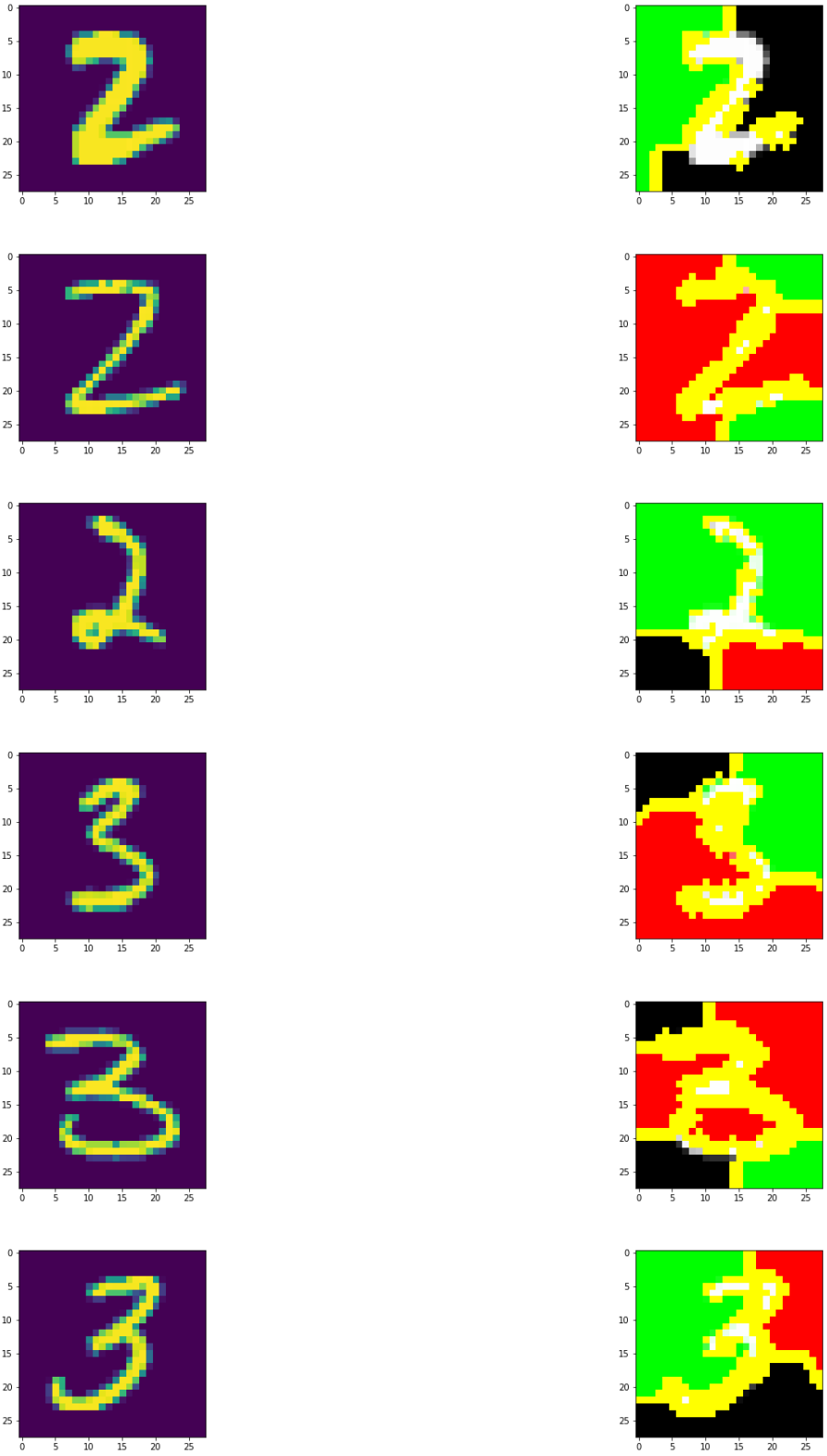


Figure 24. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

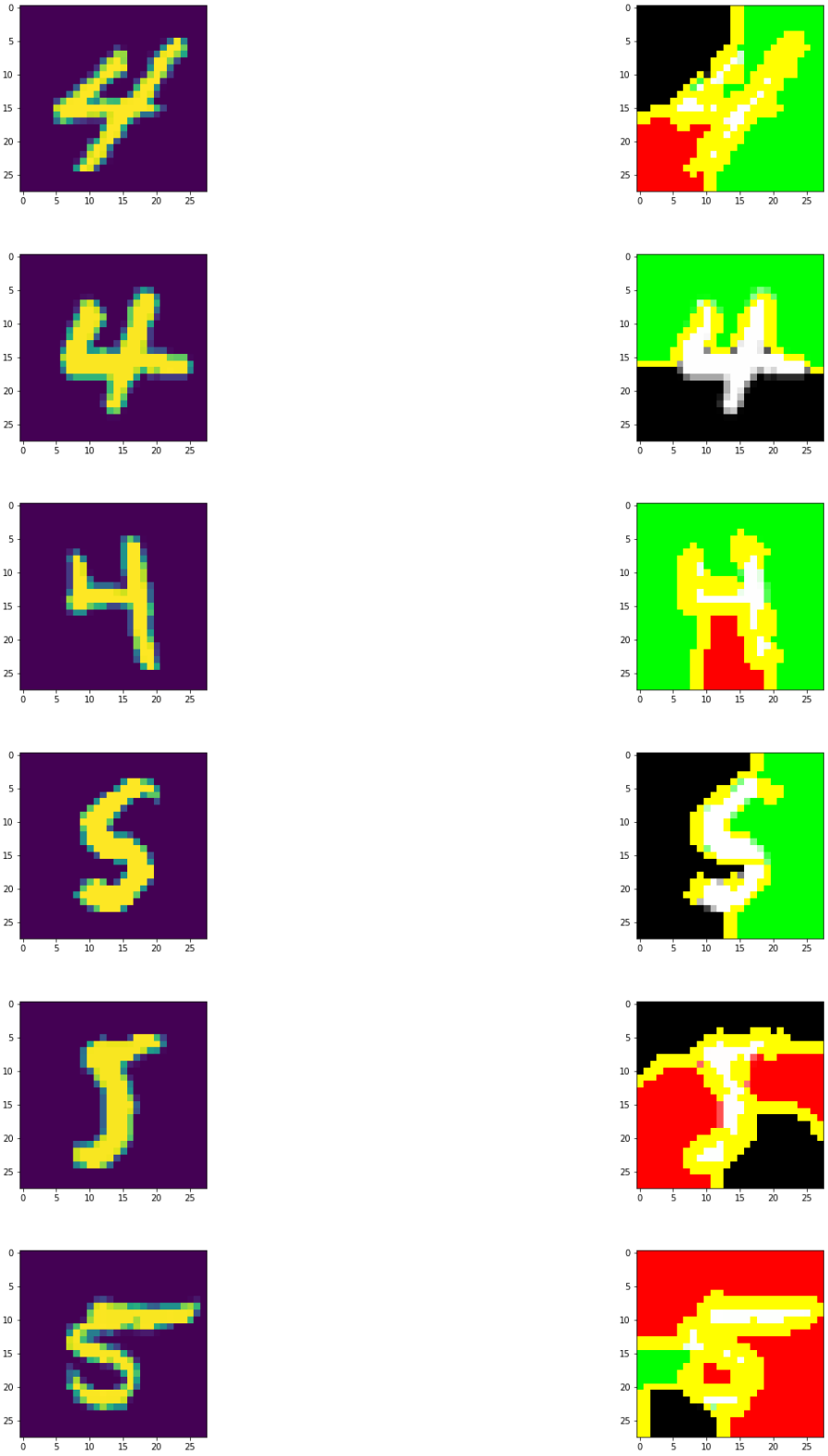


Figure 25. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

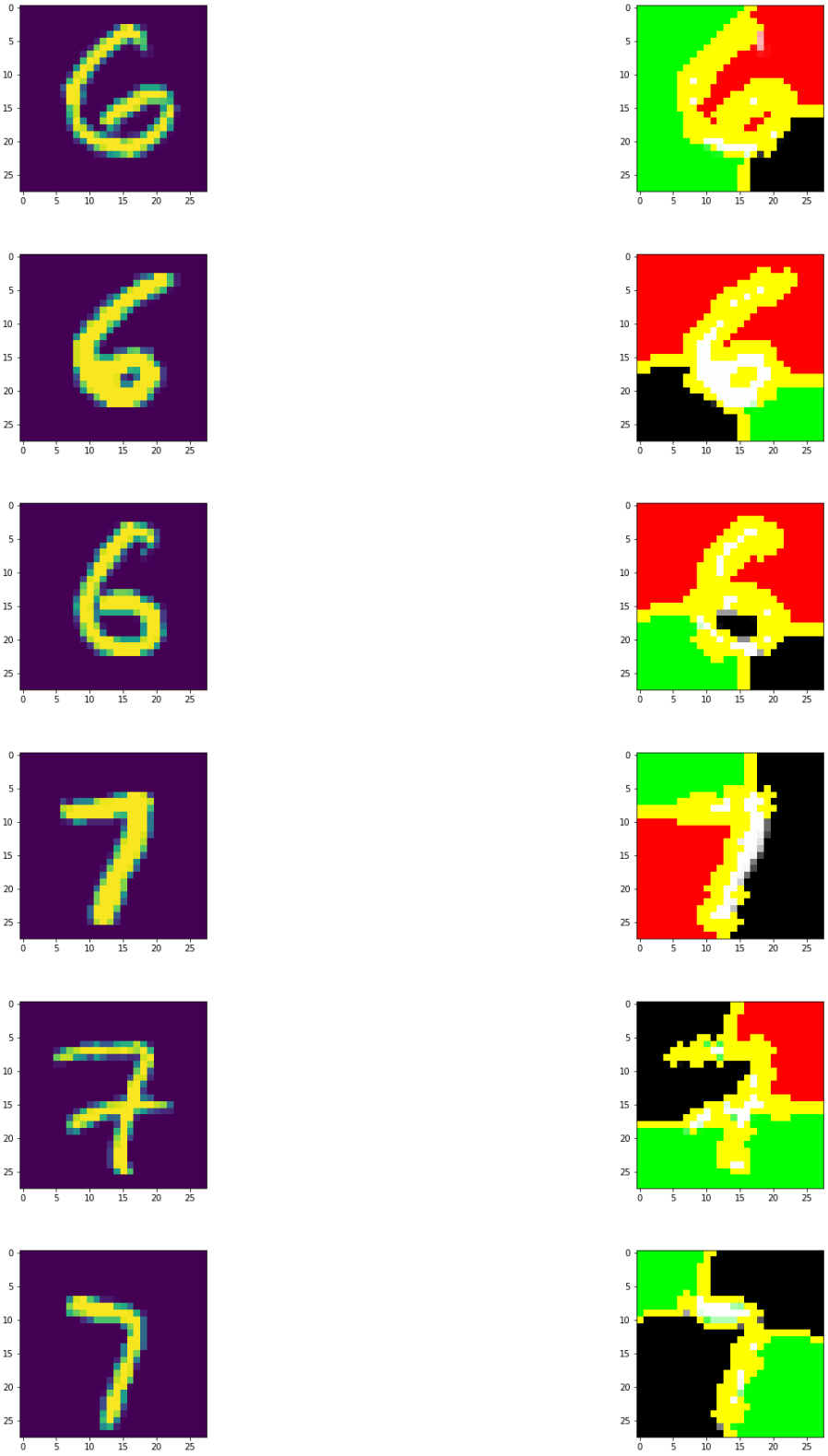


Figure 26. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

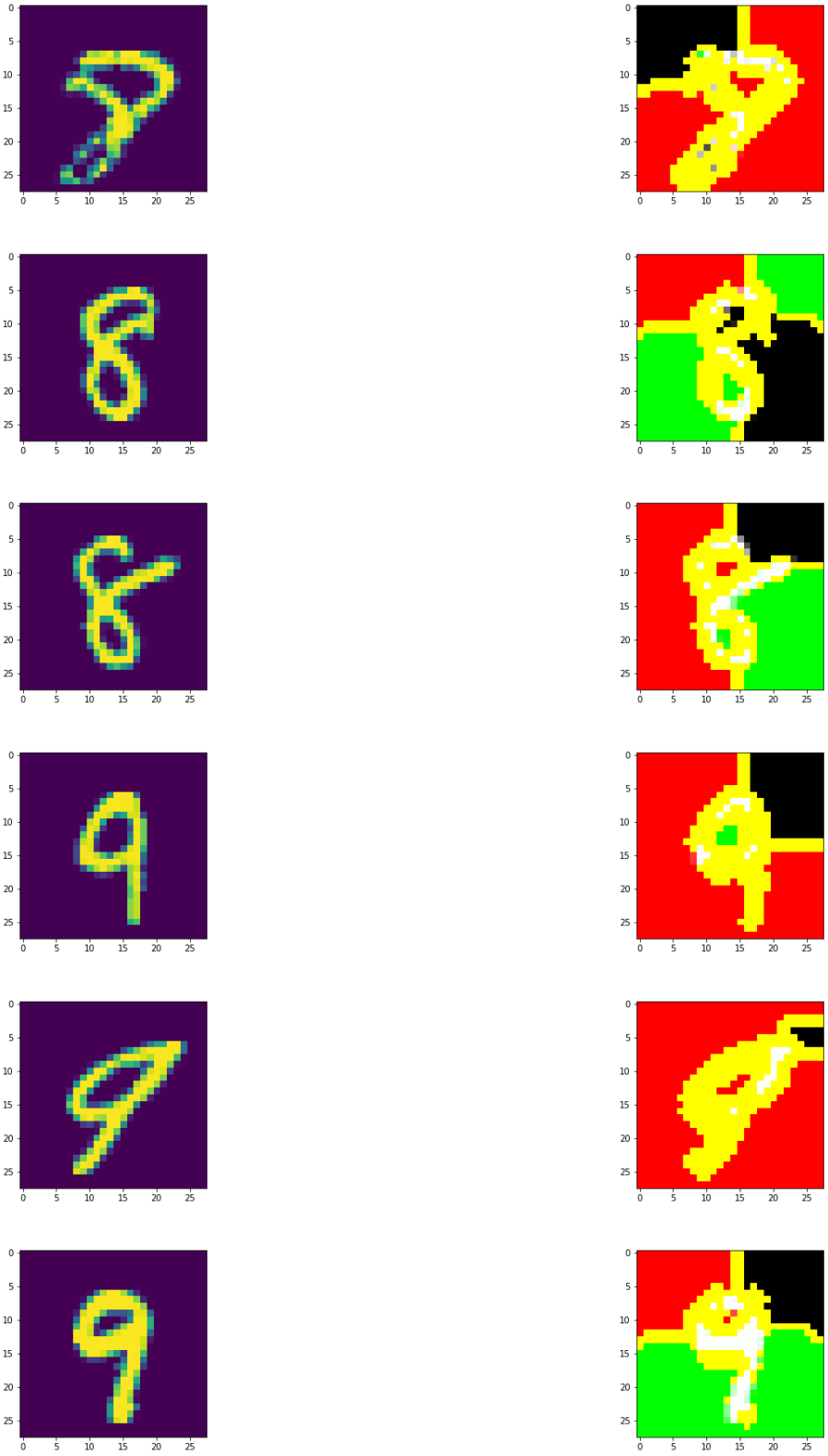


Figure 27. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

11.2 CIFAR10 LIME

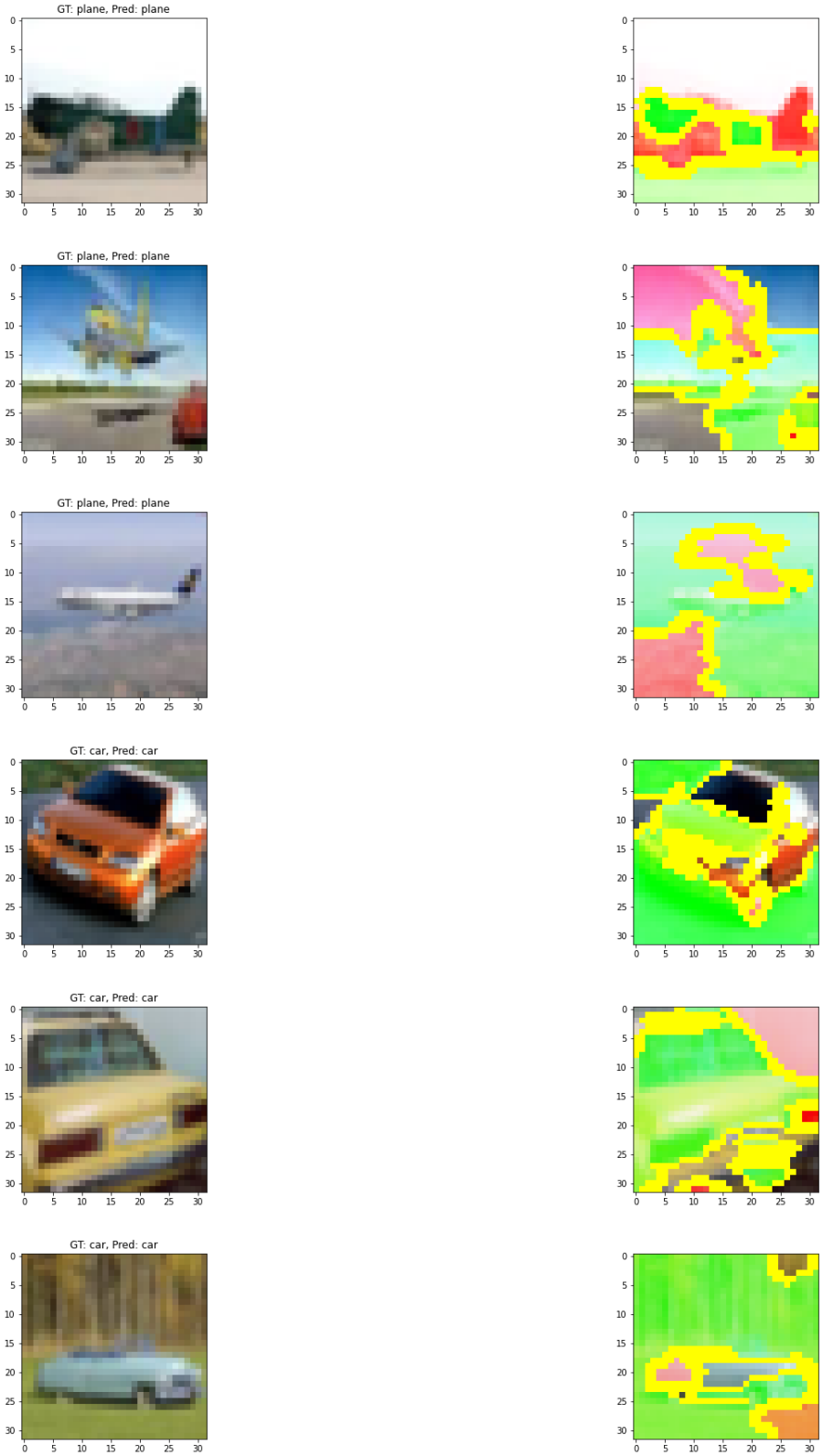


Figure 28. Original and Lime explanation of CIFAR10 images. Images in left column are original images and images in the right column are LIME explanations.

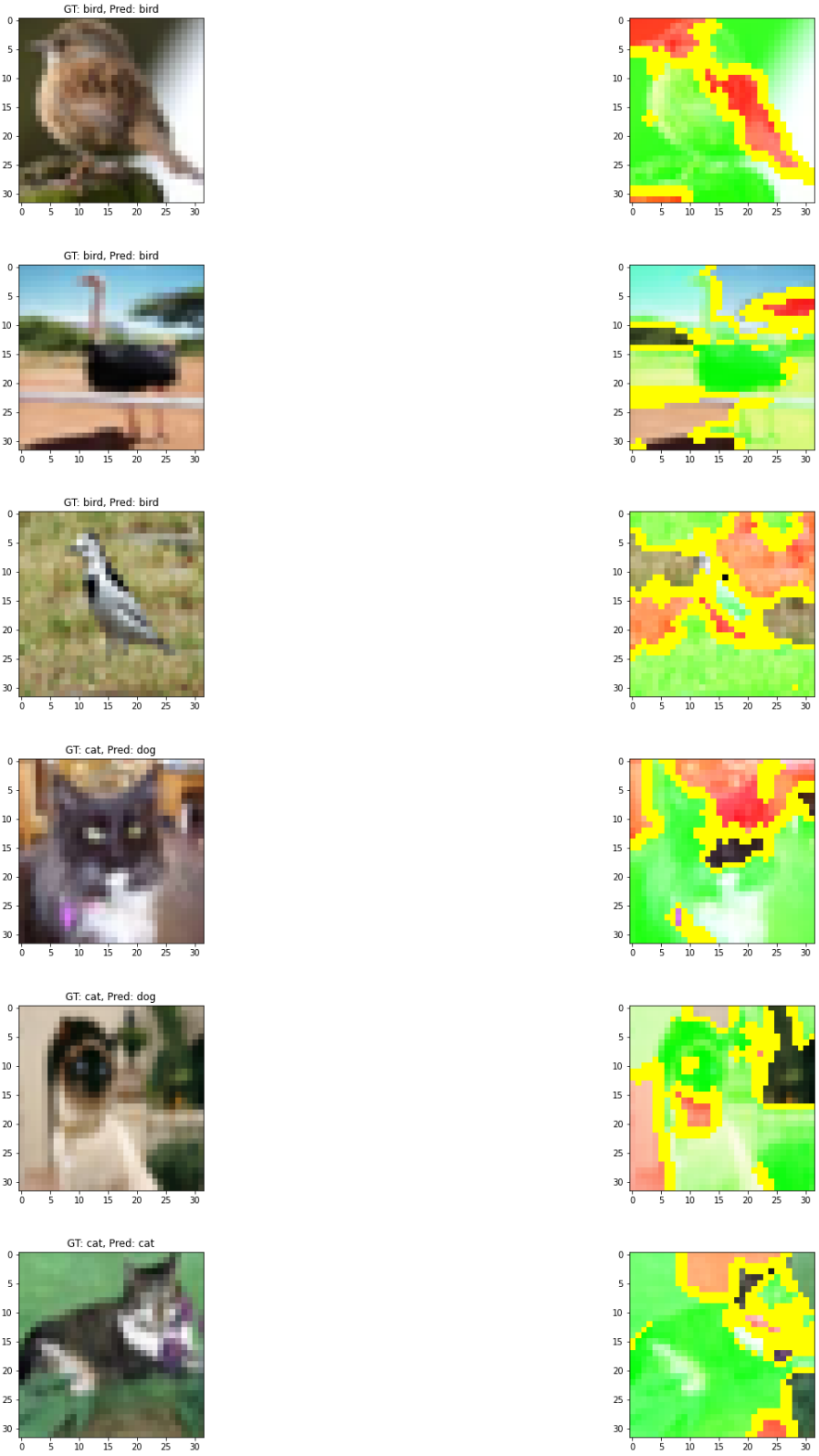


Figure 29. Original and Lime explanation of CIFAR10 images. Images in left column are original images and images in the right column are LIME explanations.

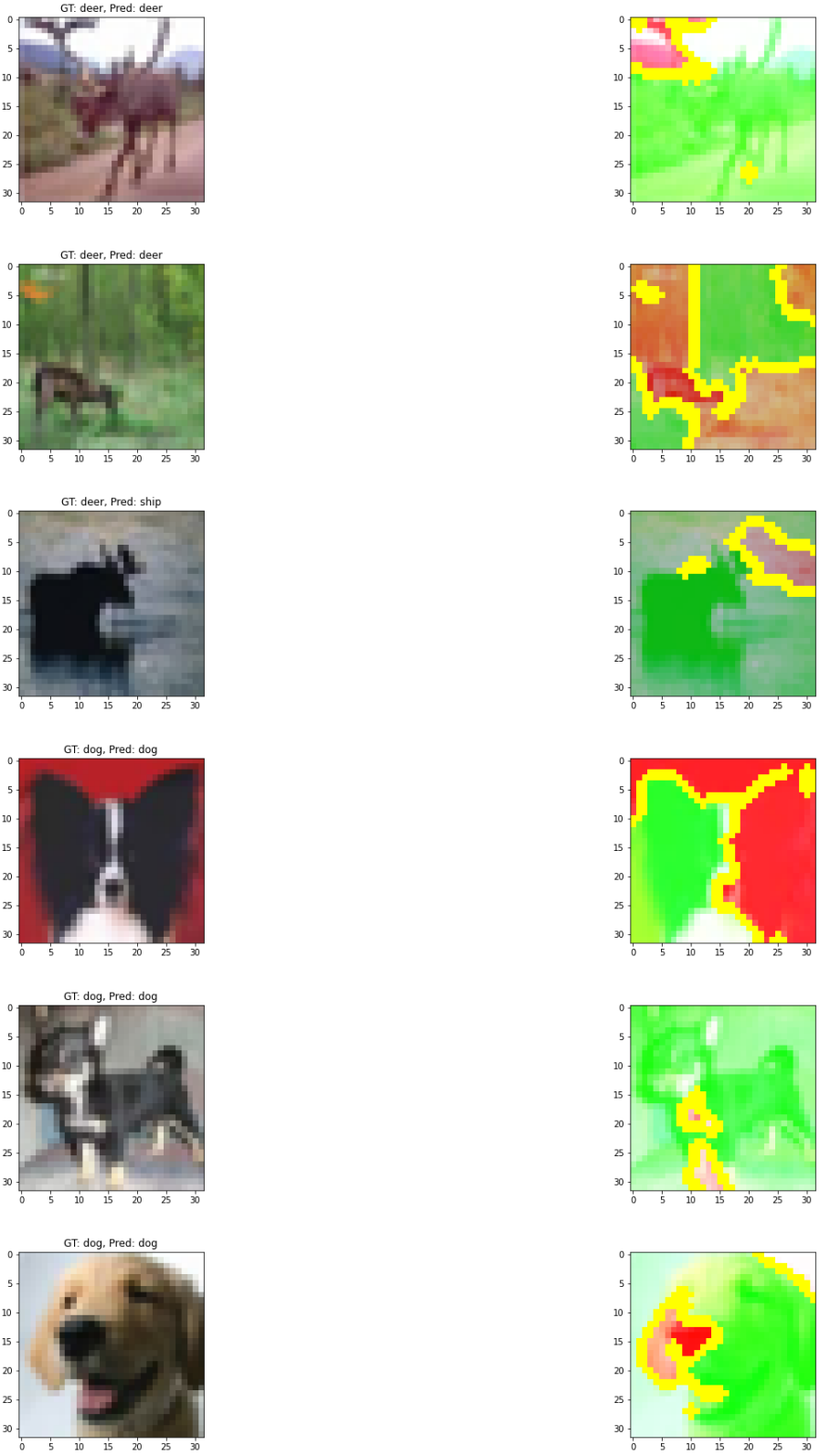


Figure 30. Original and Lime explanation of CIFAR10 images. Images in left column are original images and images in the right column are LIME explanations.

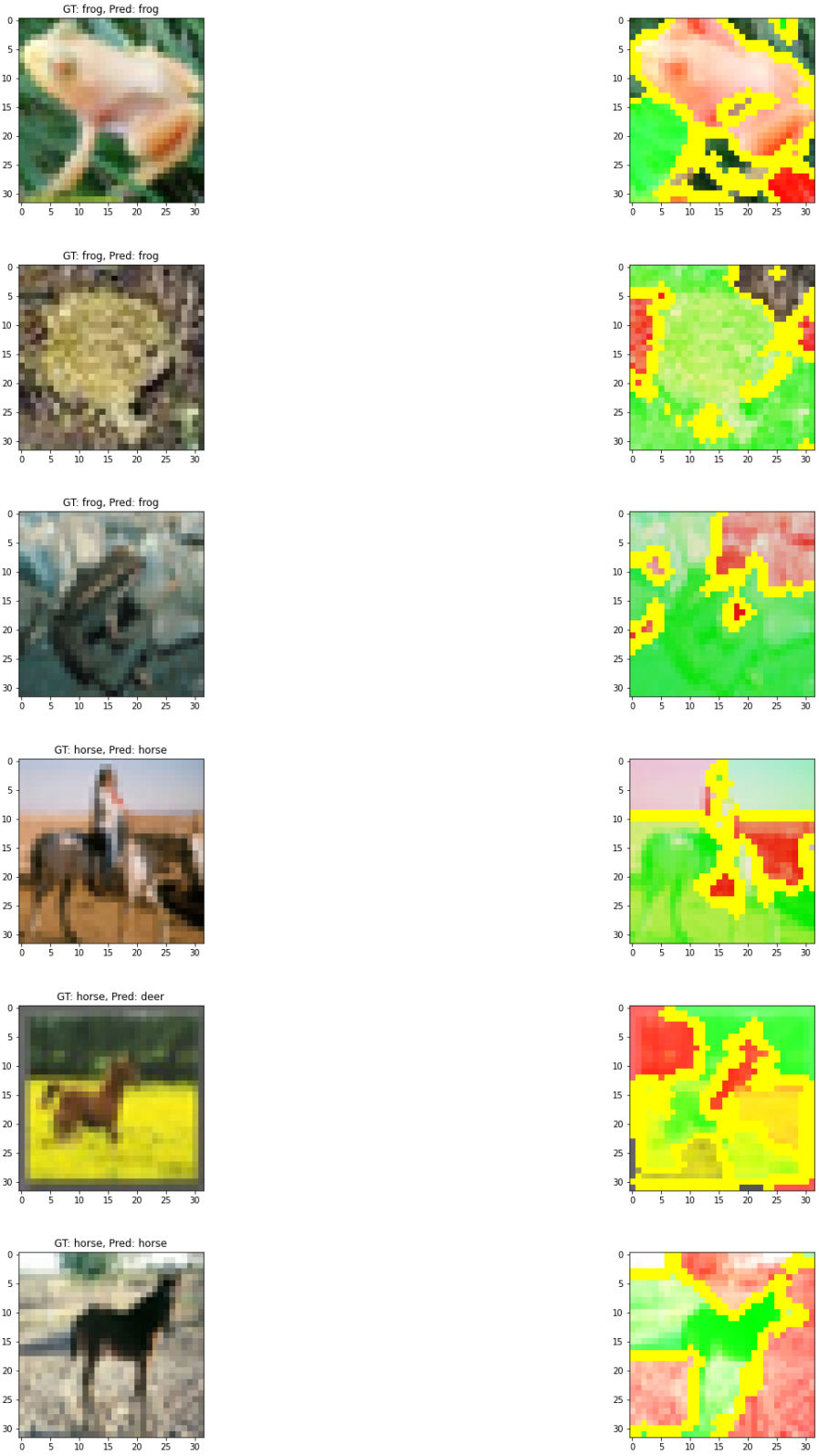


Figure 31. Original and Lime explanation of CIFAR10 images. Images in left column are original images and images in the right column are LIME explanations.

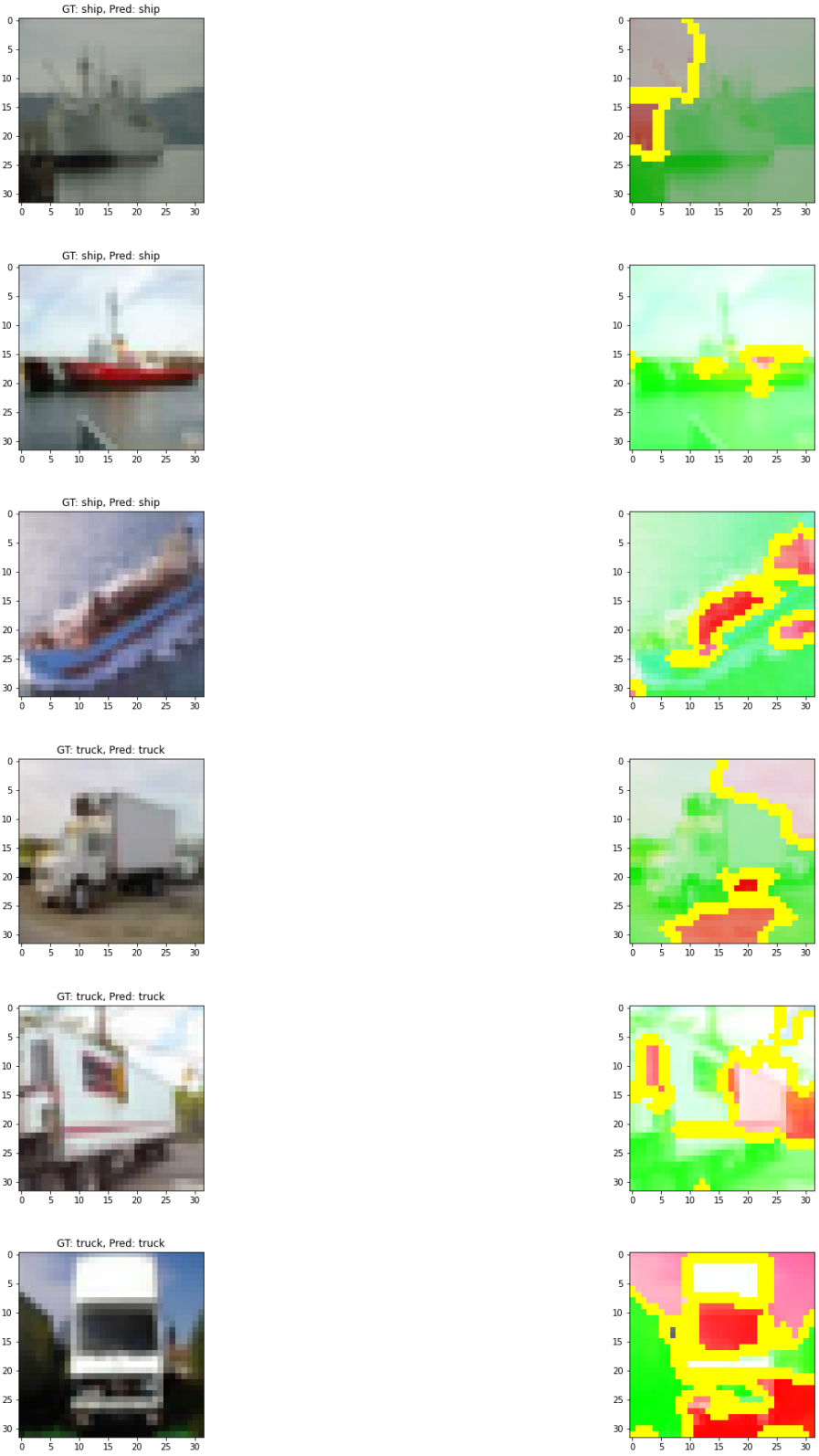


Figure 32. Original and Lime explanation of CIFAR10 images. Images in left column are original images and images in the right column are LIME explanations.

11.3 MNIST LIME BNN

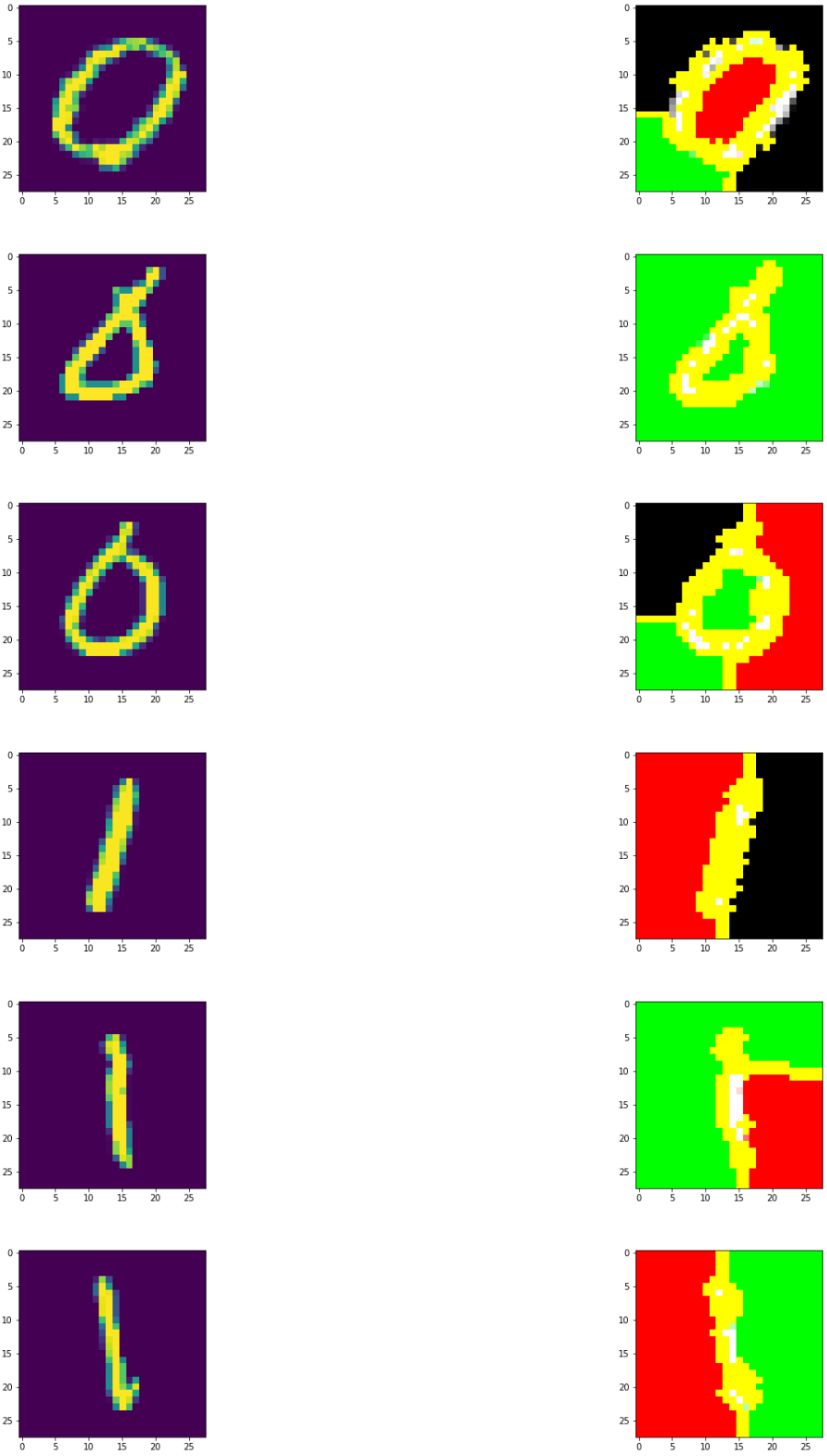


Figure 33. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

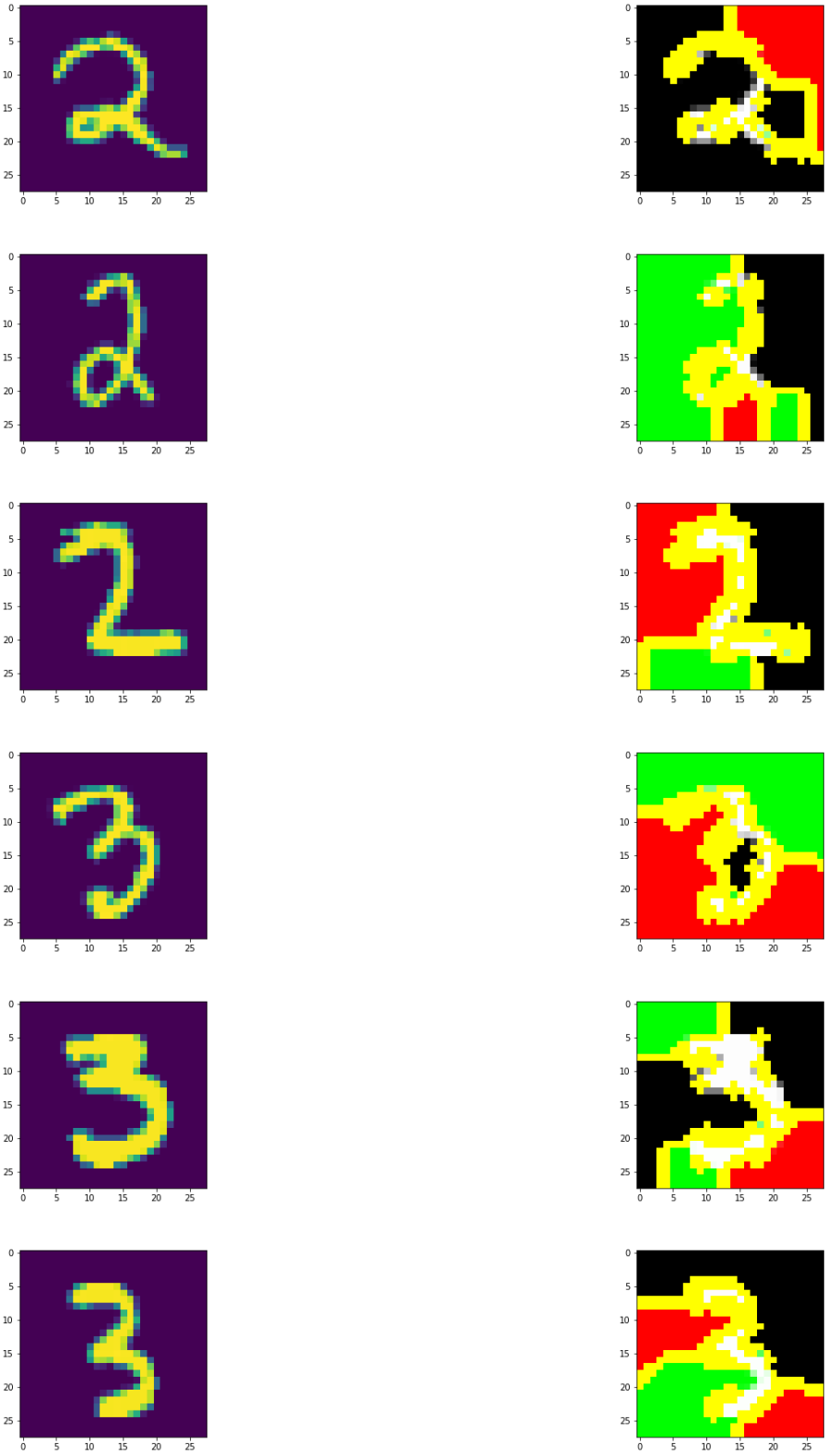


Figure 34. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

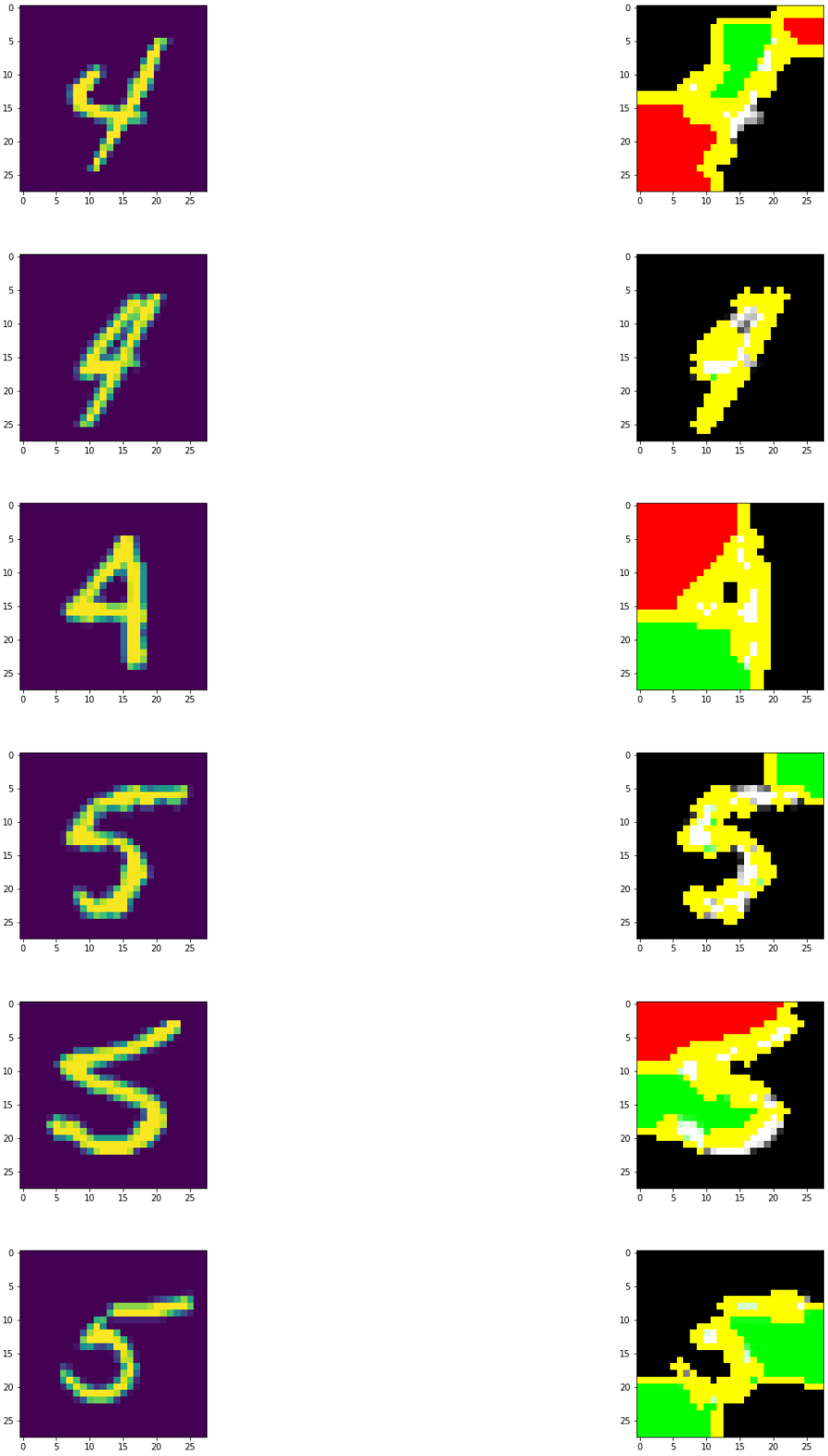


Figure 35. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

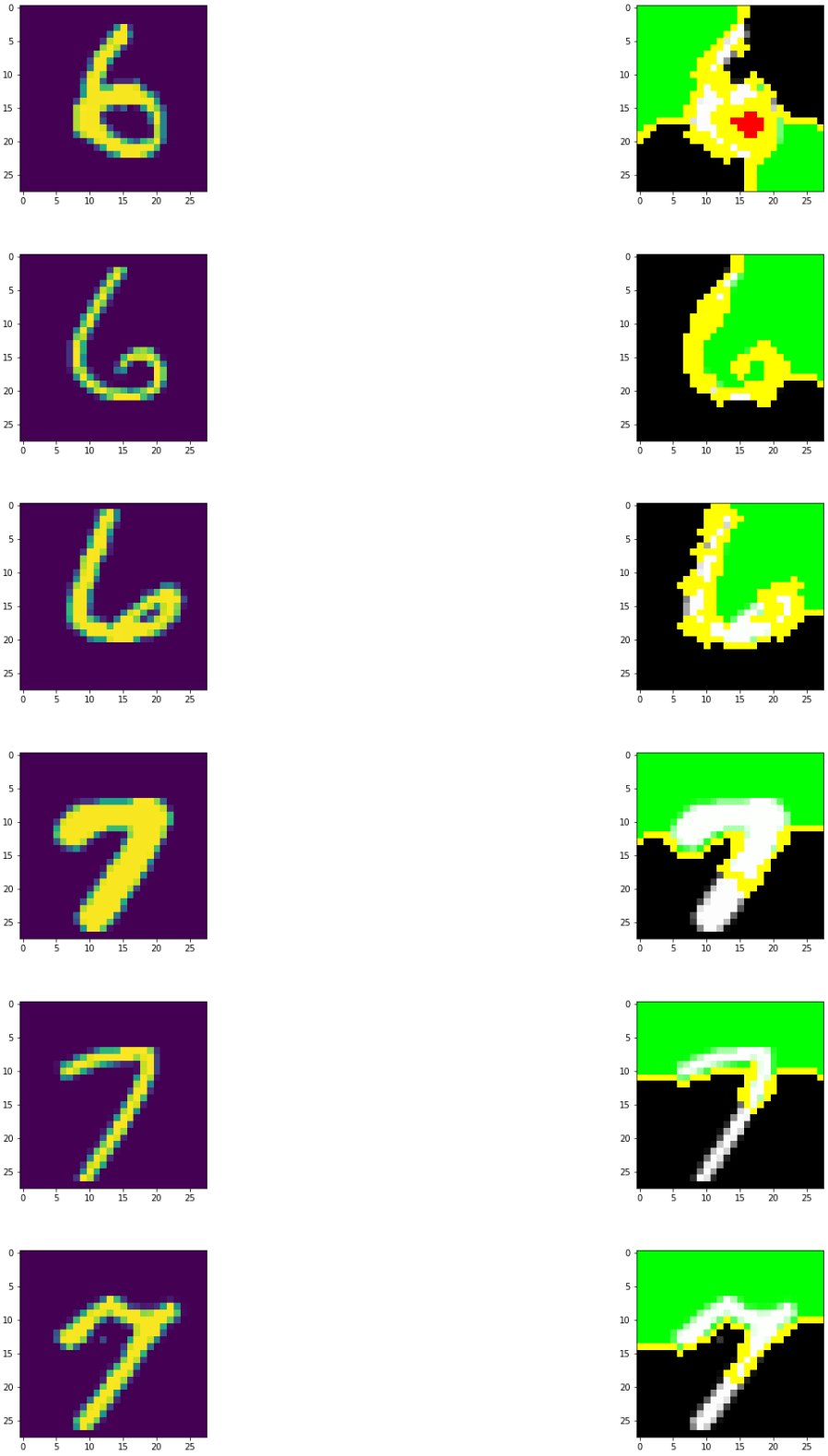


Figure 36. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

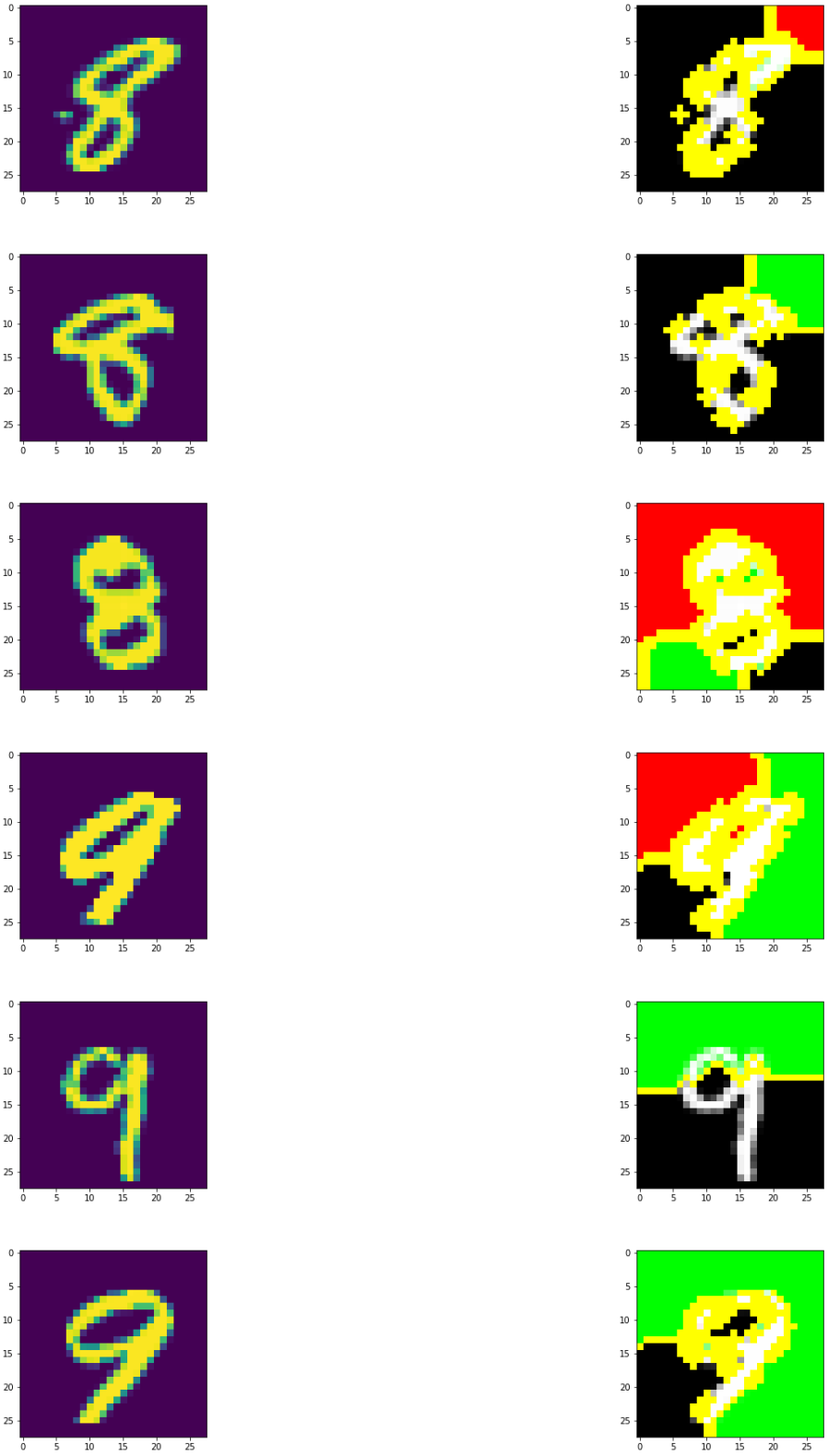
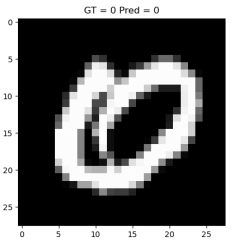


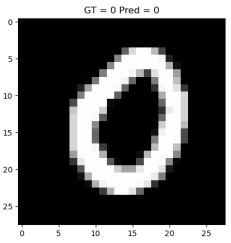
Figure 37. Original and Lime explanation of MNIST images. Images in left column are original images and images in the right column are LIME explanations.

11.4 MNIST LRP

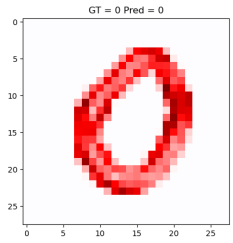


(a) Original images

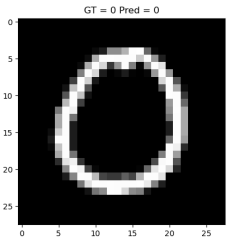
(b) LRP explanation of the image



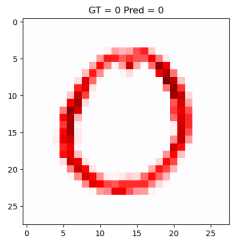
(c) Original images



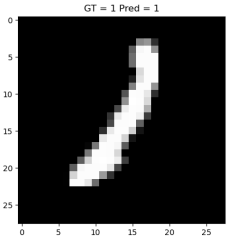
(d) LRP explanation of the image



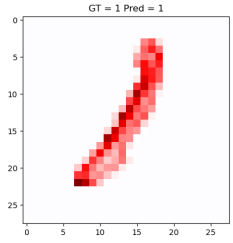
(e) Original images



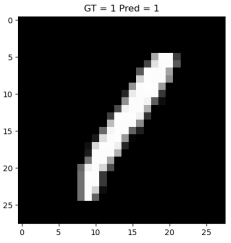
(f) LRP explanation of the image



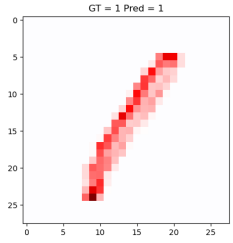
(g) Original images



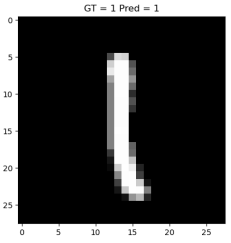
(h) LRP explanation of the image



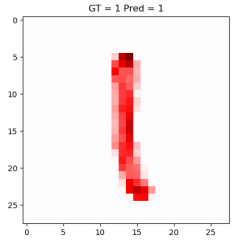
(i) Original images



(j) LRP explanation of the image

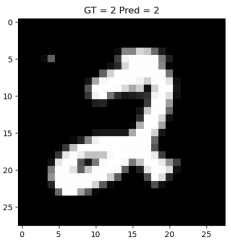


(k) Original images

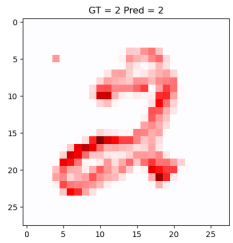


(l) LRP explanation of the image

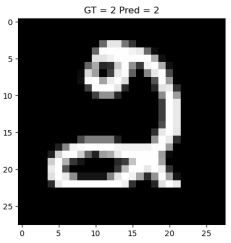
Figure 38. Original and LRP explanation of MNIST images



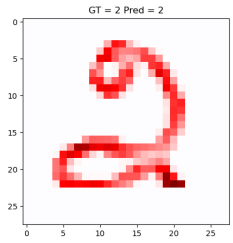
(a) Original images



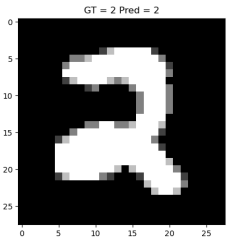
(b) LRP explanation of the image



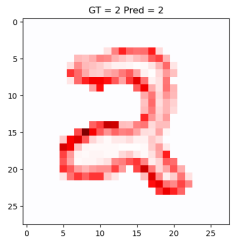
(c) Original images



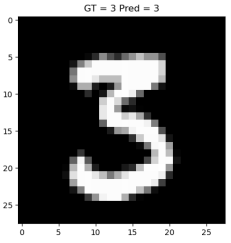
(d) LRP explanation of the image



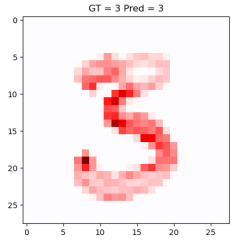
(e) Original images



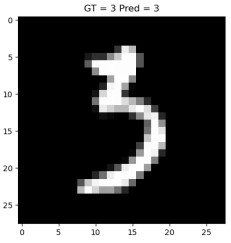
(f) LRP explanation of the image



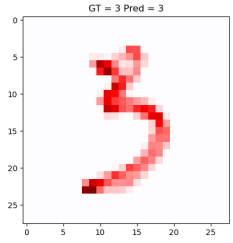
(g) Original images



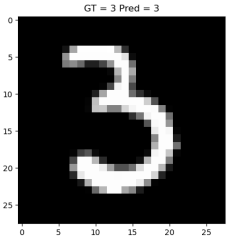
(h) LRP explanation of the image



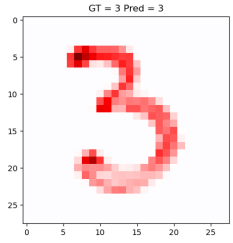
(i) Original images



(j) LRP explanation of the image

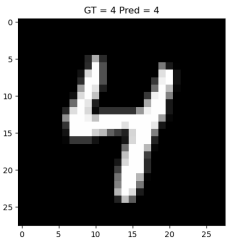


(k) Original images

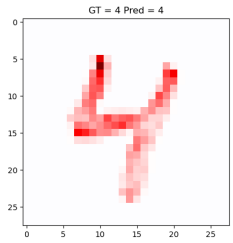


(l) LRP explanation of the image

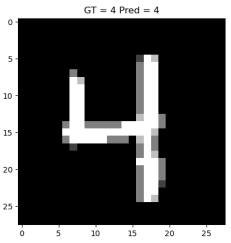
Figure 39. Original and LRP explanation of MNIST images



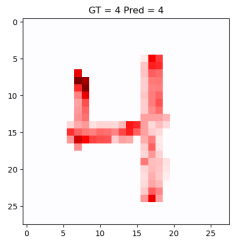
(a) Original images



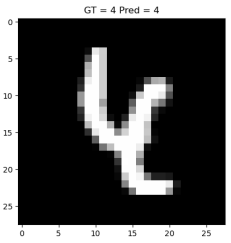
(b) LRP explanation of the image



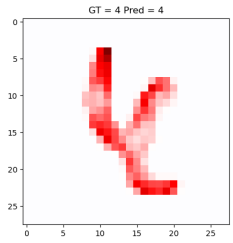
(c) Original images



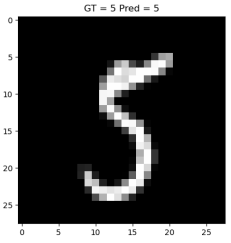
(d) LRP explanation of the image



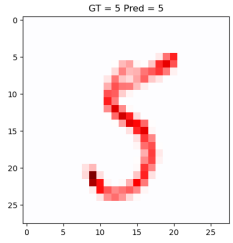
(e) Original images



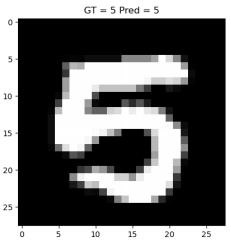
(f) LRP explanation of the image



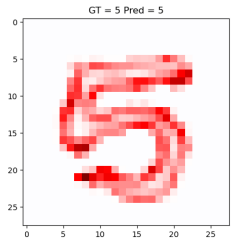
(g) Original images



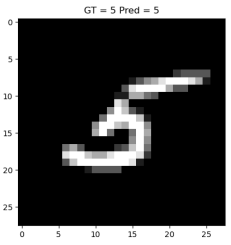
(h) LRP explanation of the image



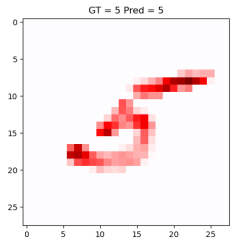
(i) Original images



(j) LRP explanation of the image

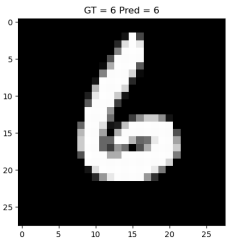


(k) Original images

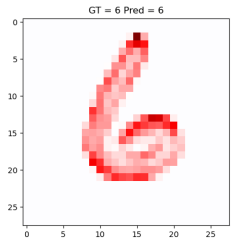


(l) LRP explanation of the image

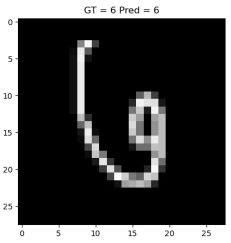
Figure 40. Original and LRP explanation of MNIST images



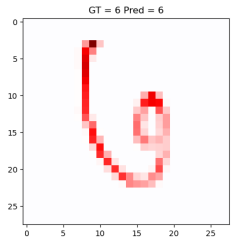
(a) Original images



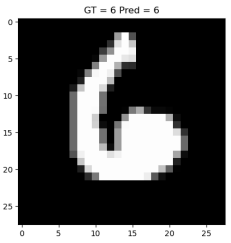
(b) LRP explanation of the image



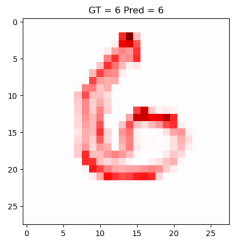
(c) Original images



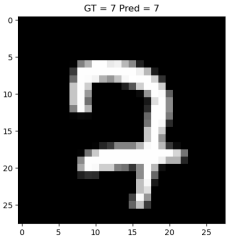
(d) LRP explanation of the image



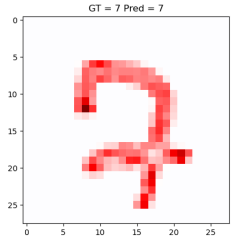
(e) Original images



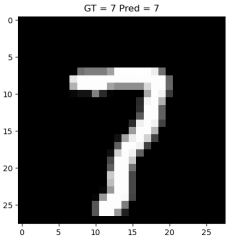
(f) LRP explanation of the image



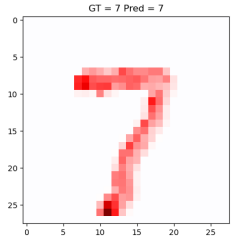
(g) Original images



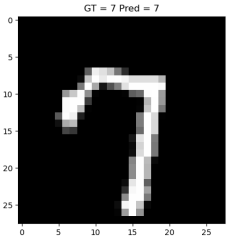
(h) LRP explanation of the image



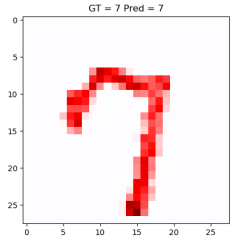
(i) Original images



(j) LRP explanation of the image

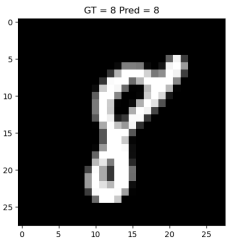


(k) Original images

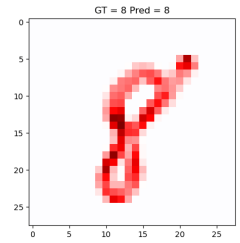


(l) LRP explanation of the image

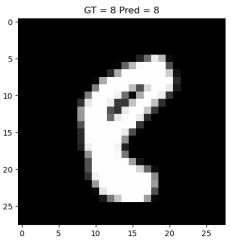
Figure 41. Original and LRP explanation of MNIST images



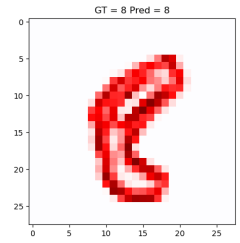
(a) Original images



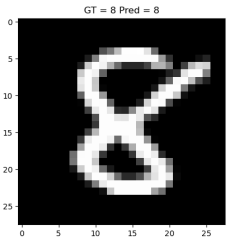
(b) LRP explanation of the image



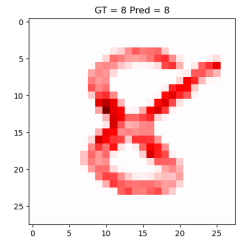
(c) Original images



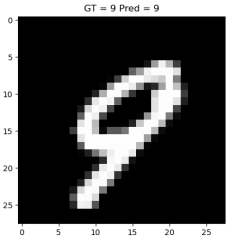
(d) LRP explanation of the image



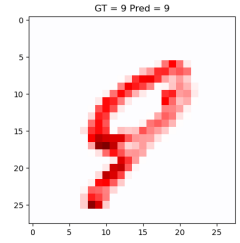
(e) Original images



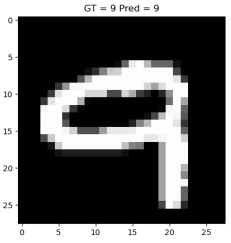
(f) LRP explanation of the image



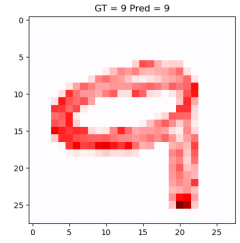
(g) Original images



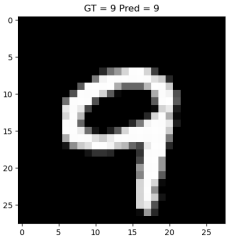
(h) LRP explanation of the image



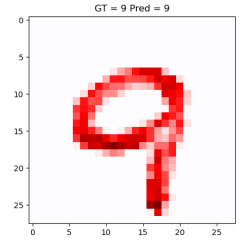
(i) Original images



(j) LRP explanation of the image



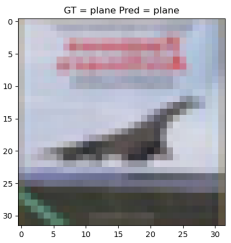
(k) Original images



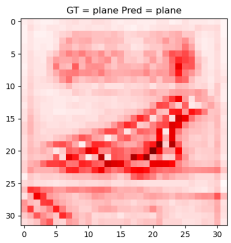
(l) LRP explanation of the image

Figure 42. Original and LRP explanation of MNIST images

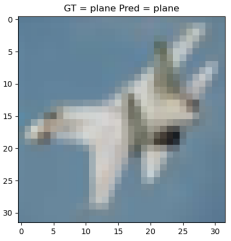
11.5 CIFAR10 LRP



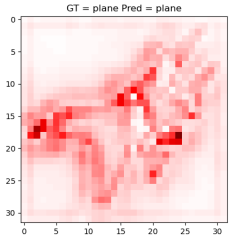
(a) Original images



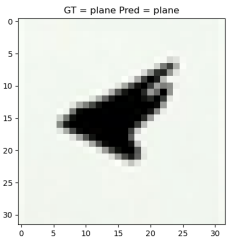
(b) LRP explanation of the image



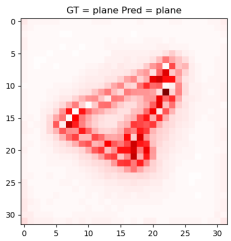
(c) Original images



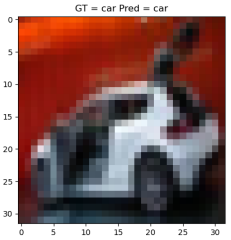
(d) LRP explanation of the image



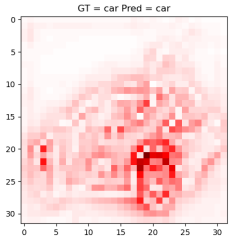
(e) Original images



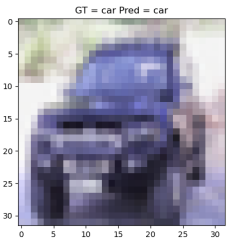
(f) LRP explanation of the image



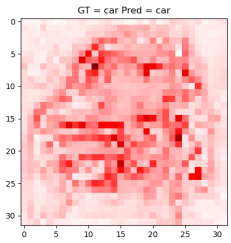
(g) Original images



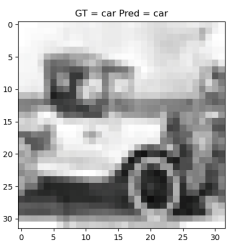
(h) LRP explanation of the image



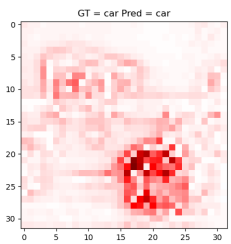
(i) Original images



(j) LRP explanation of the image

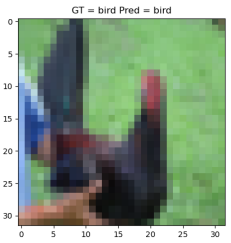


(k) Original images

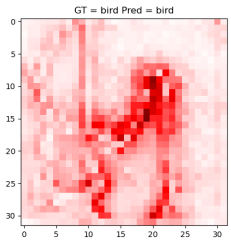


(l) LRP explanation of the image

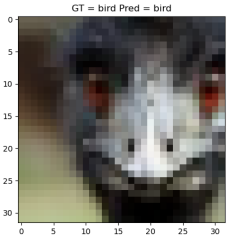
Figure 43. Original and LRP explanation of CIFAR10 images



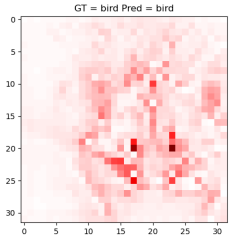
(a) Original images



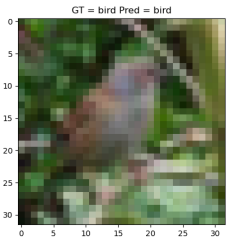
(b) LRP explanation of the image



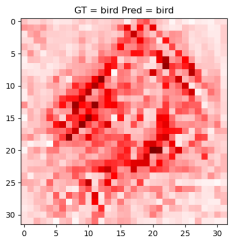
(c) Original images



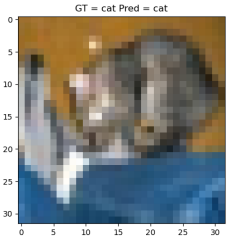
(d) LRP explanation of the image



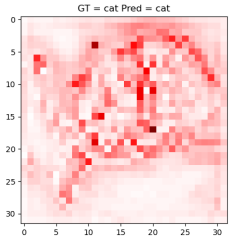
(e) Original images



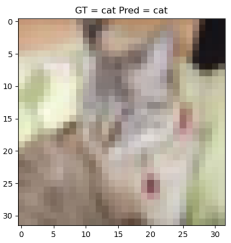
(f) LRP explanation of the image



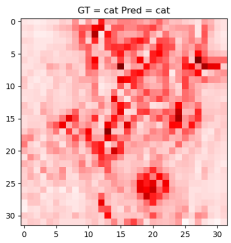
(g) Original images



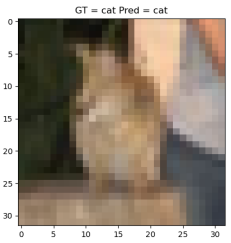
(h) LRP explanation of the image



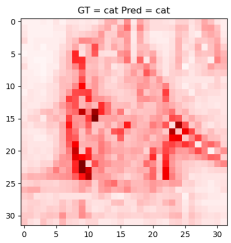
(i) Original images



(j) LRP explanation of the image

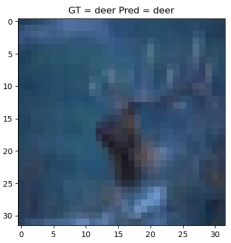


(k) Original images

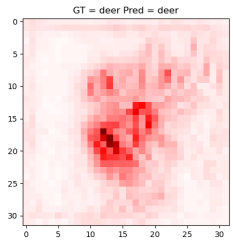


(l) LRP explanation of the image

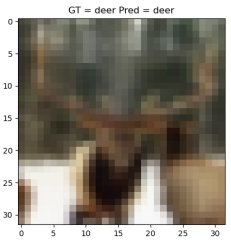
Figure 44. Original and LRP explanation of CIFAR10 images



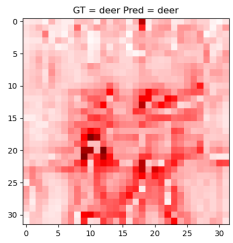
(a) Original images



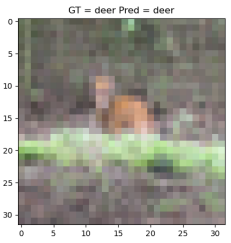
(b) LRP explanation of the image



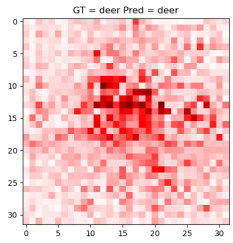
(c) Original images



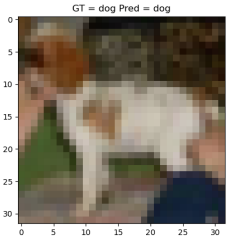
(d) LRP explanation of the image



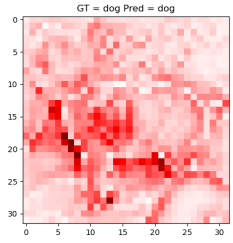
(e) Original images



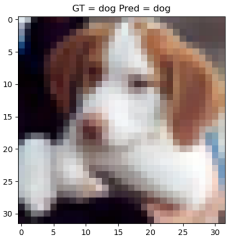
(f) LRP explanation of the image



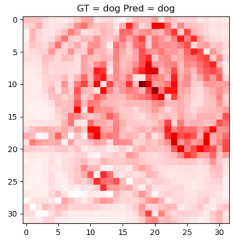
(g) Original images



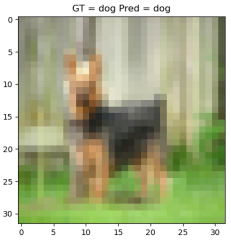
(h) LRP explanation of the image



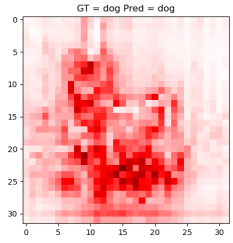
(i) Original images



(j) LRP explanation of the image

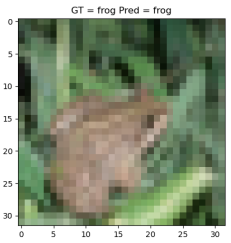


(k) Original images

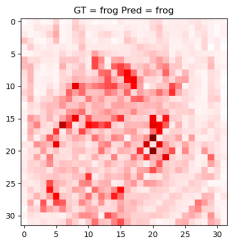


(l) LRP explanation of the image

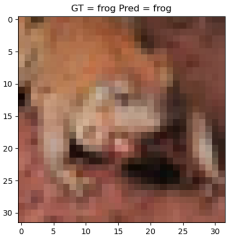
Figure 45. Original and LRP explanation of CIFAR10 images



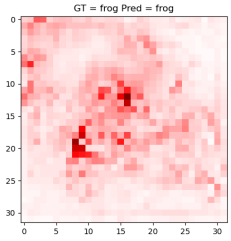
(a) Original images



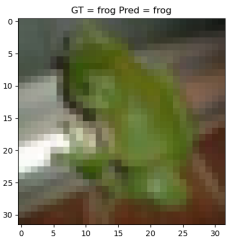
(b) LRP explanation of the image



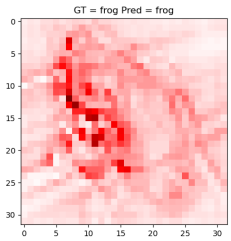
(c) Original images



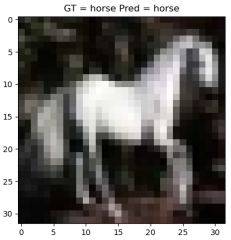
(d) LRP explanation of the image



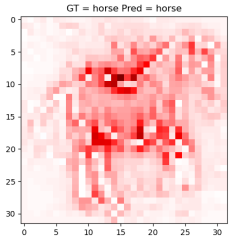
(e) Original images



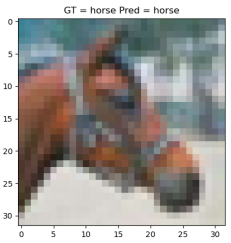
(f) LRP explanation of the image



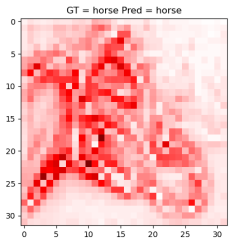
(g) Original images



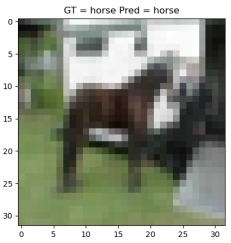
(h) LRP explanation of the image



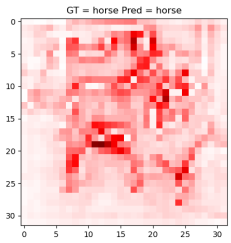
(i) Original images



(j) LRP explanation of the image

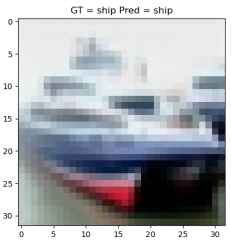


(k) Original images



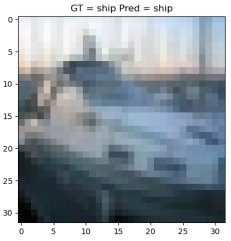
(l) LRP explanation of the image

Figure 46. Original and LRP explanation of CIFAR10 images

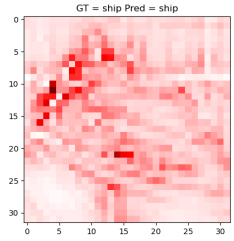


(a) Original images

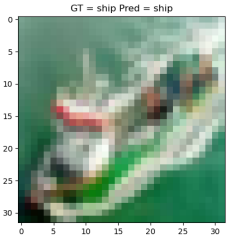
(b) LRP explanation of the image



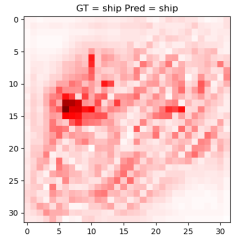
(c) Original images



(d) LRP explanation of the image

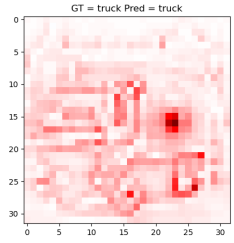


(e) Original images

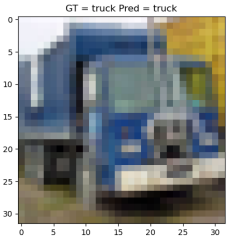


(f) LRP explanation of the image

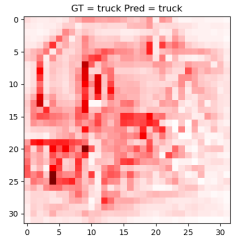
(g) Original images



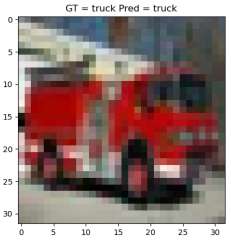
(h) LRP explanation of the image



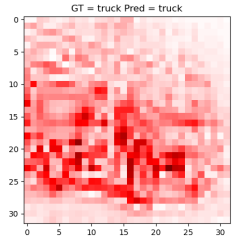
(i) Original images



(j) LRP explanation of the image



(k) Original images



(l) LRP explanation of the image

Figure 47. Original and LRP explanation of CIFAR10 images

11.6 CIFAR10 KernelShap

11.6.1 Cat Interpreation Samples

Here you can see interpretation samples of dog vs cat images which you can see exactly why a particular image has been classified as cat. After we will try this on dog images.

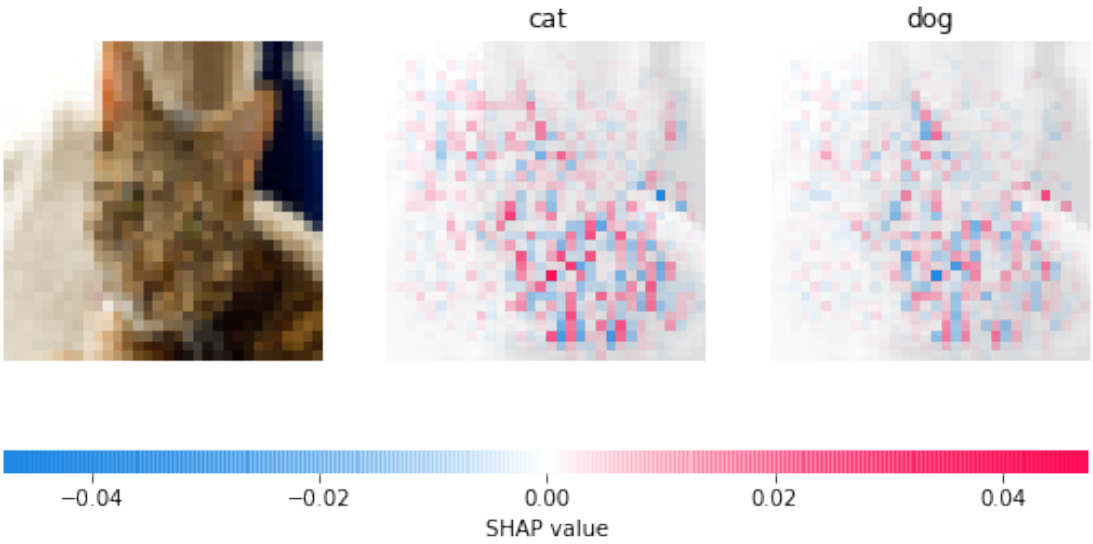


Figure 48. KernelShap Interpretation Sample Cat Image 1

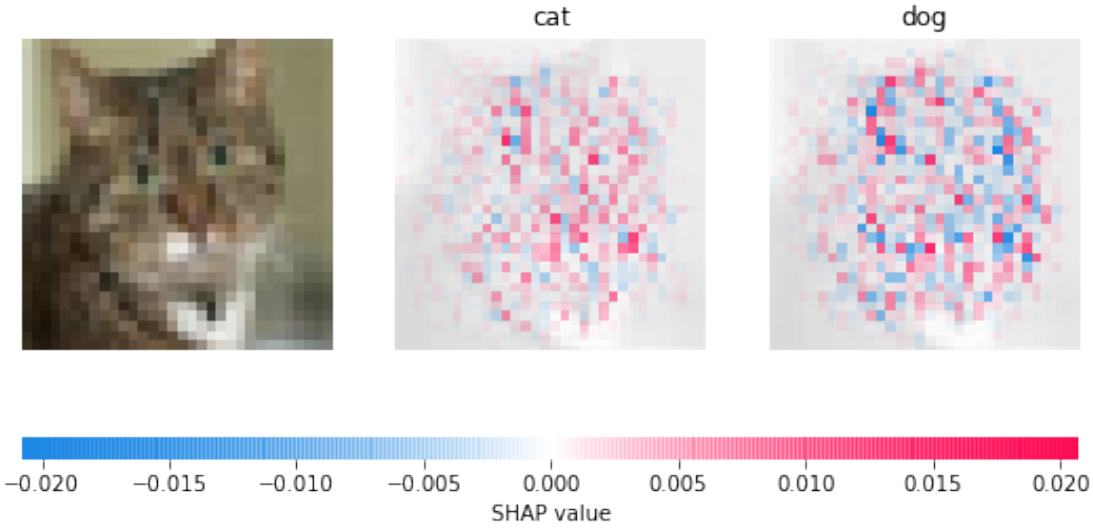


Figure 49. KernelShap Interpretation Sample Cat Image 2

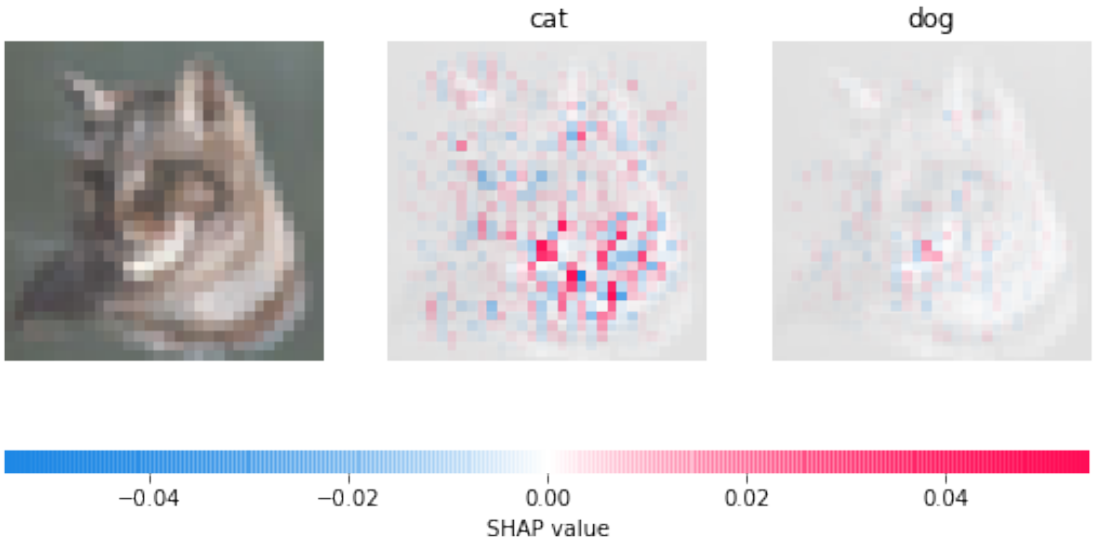


Figure 50. KernelShap Interpretation Sample Cat Image 3

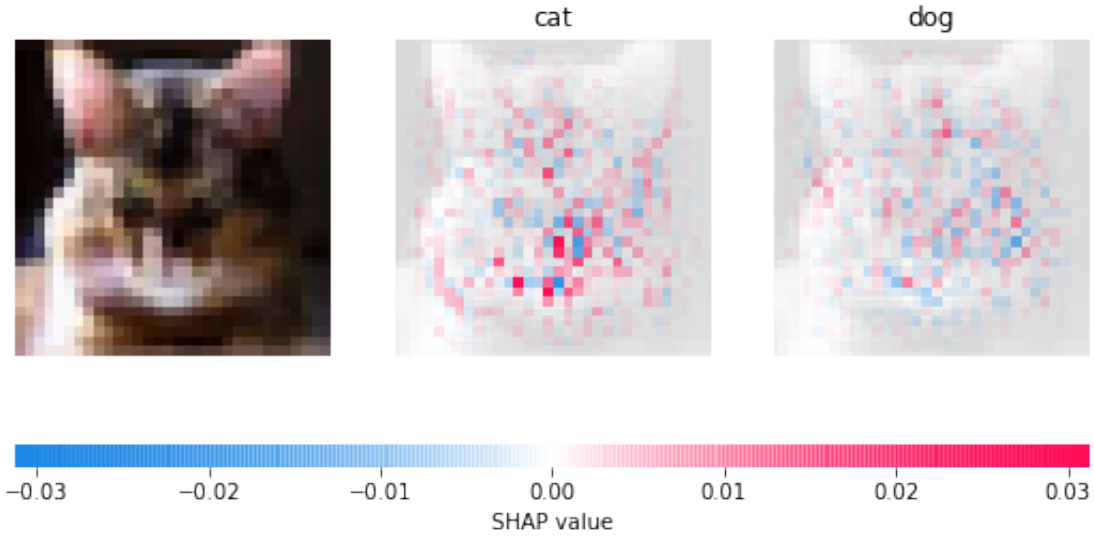


Figure 51. KernelShap Interpretation Sample Cat Image 4

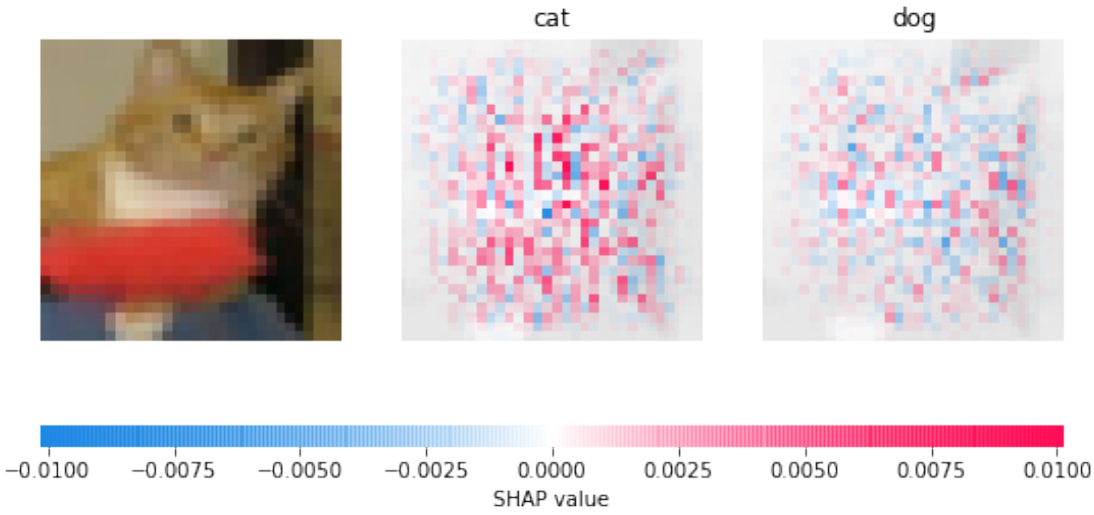


Figure 52. KernelShap Interpretation Sample Cat Image 5

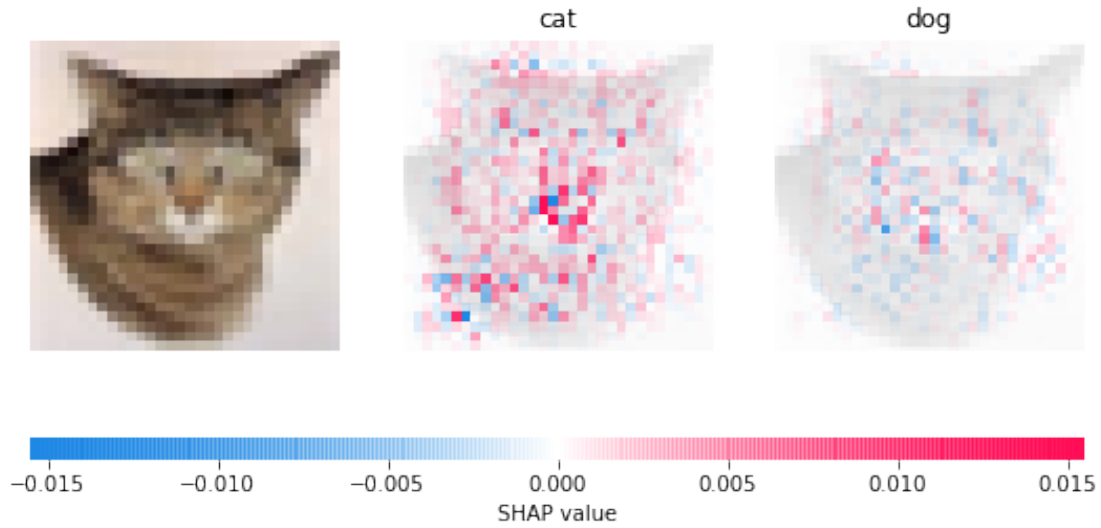


Figure 53. KernelShap Interpretation Sample Cat Image 6

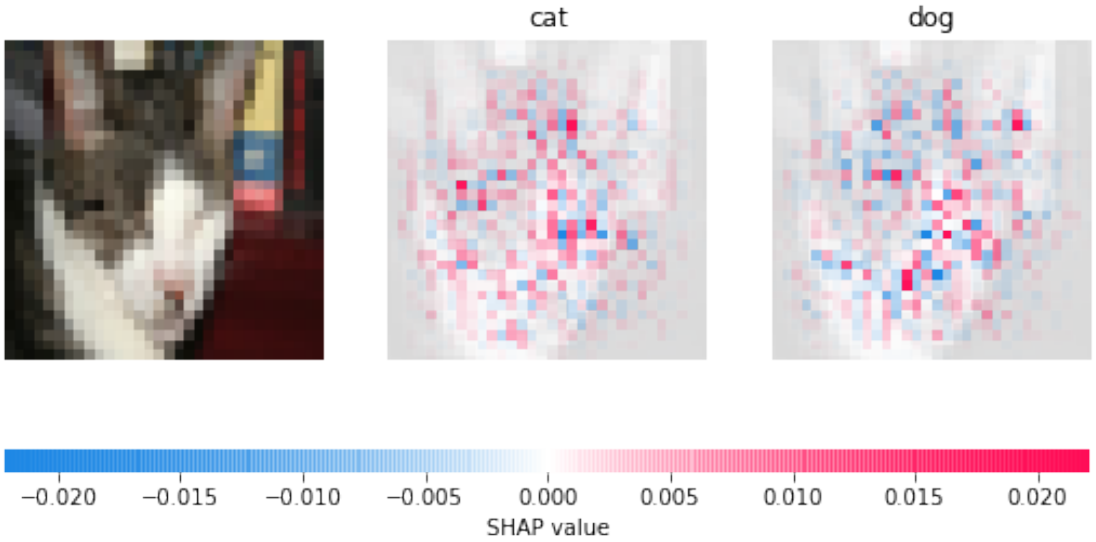


Figure 54. KernelShap Interpretation Sample Cat Image 7

11.6.2 Dog Interpretation samples

Here you can see interpretation samples of dog vs cat images which you can see exactly why a particular image has been classified as dog.

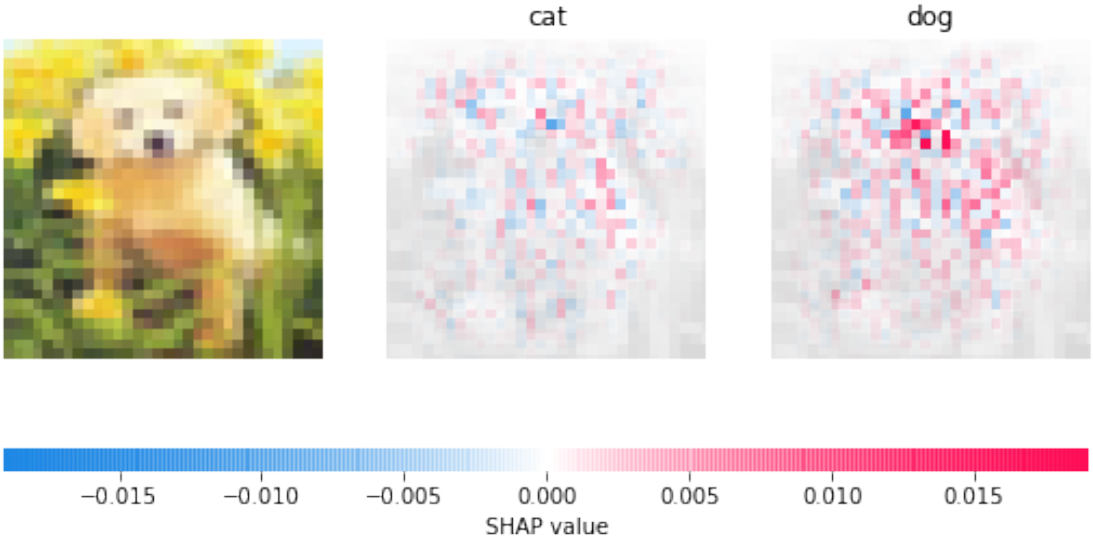


Figure 55. KernelShap Interpretation Sample Dog Image 1

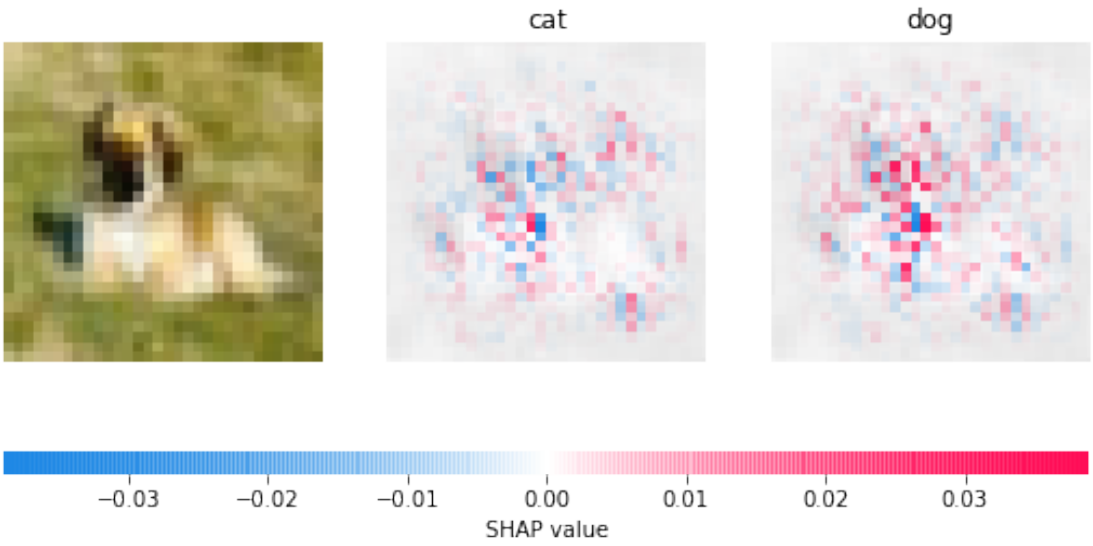


Figure 56. KernelShap Interpretation Sample Dog Image 2

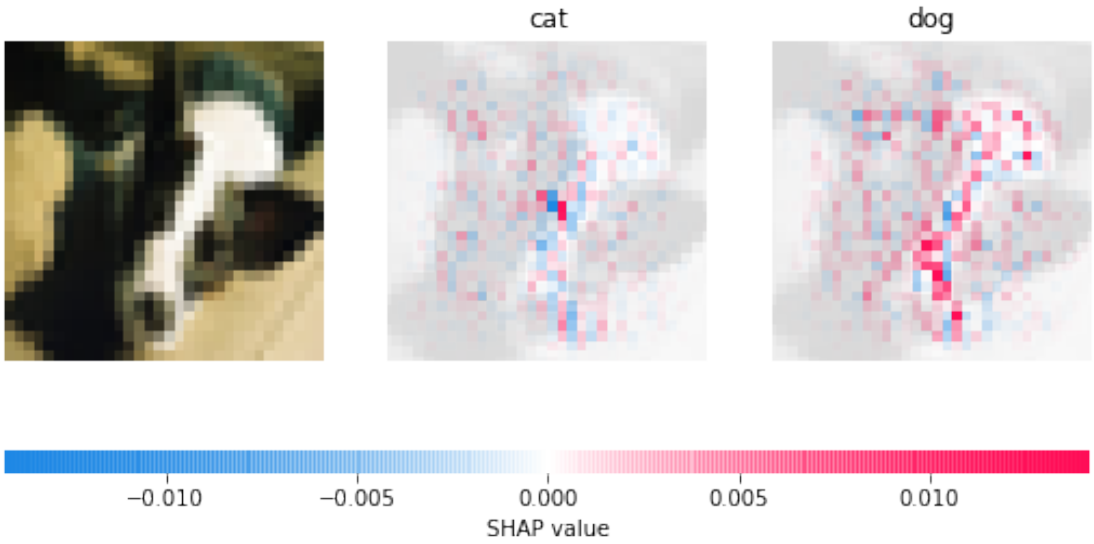


Figure 57. Kernel Shap Interpretation Sample Dog Image 3

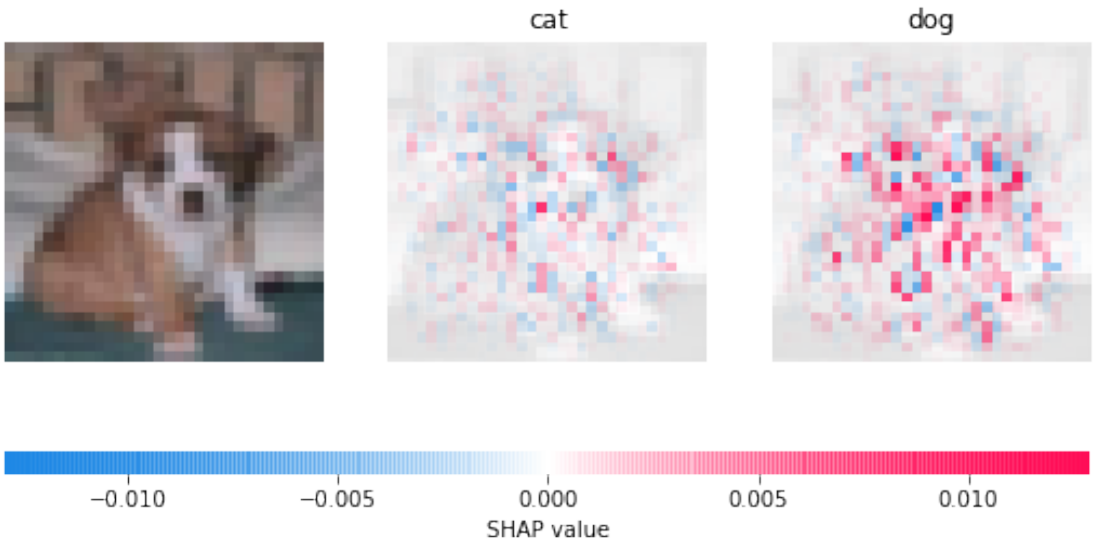


Figure 58. Kernel Shap Interpretation Sample Dog Image 4

11.6.3 Horse Interpretation Samples

Here you can see interpretation samples of horse vs deer images which you can see exactly why a particular image has been classified as horse.

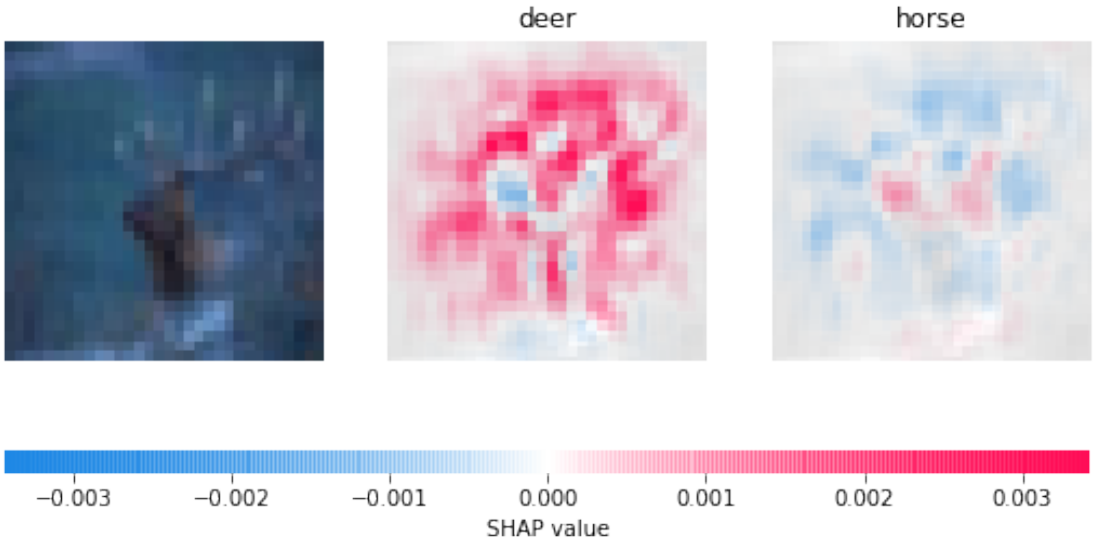


Figure 59. Kernel Shap Interpretation Sample Horse Image 1

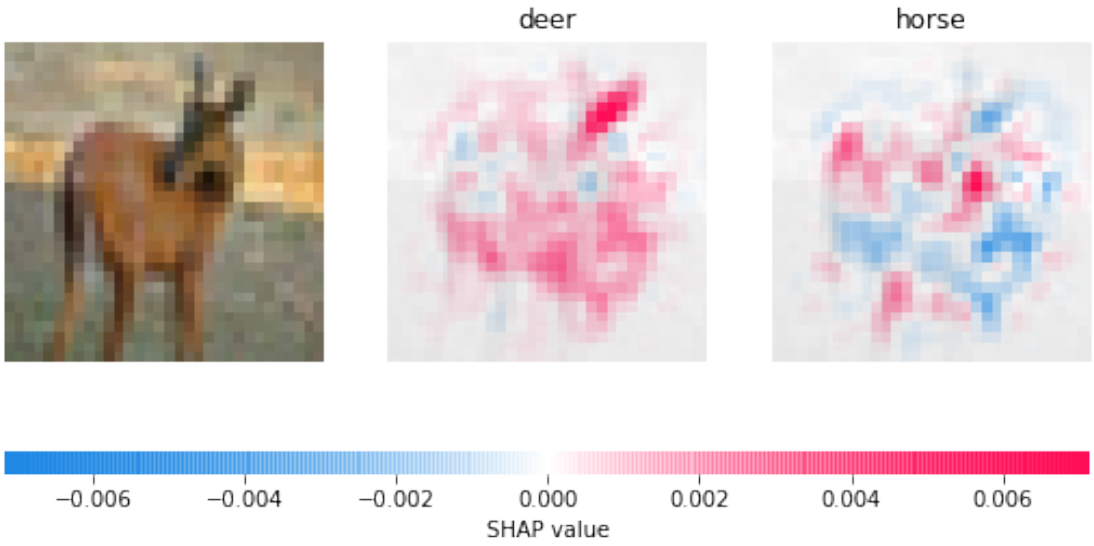


Figure 60. Kernel Shap Interpretation Sample Horse Image 2

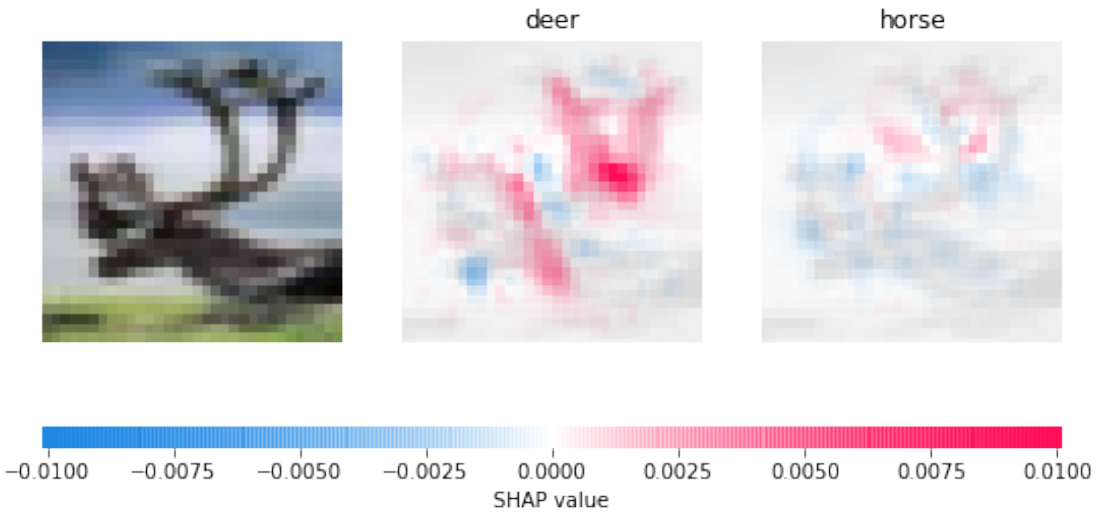


Figure 61. Kernel Shap Interpretation Sample Horse Image 3

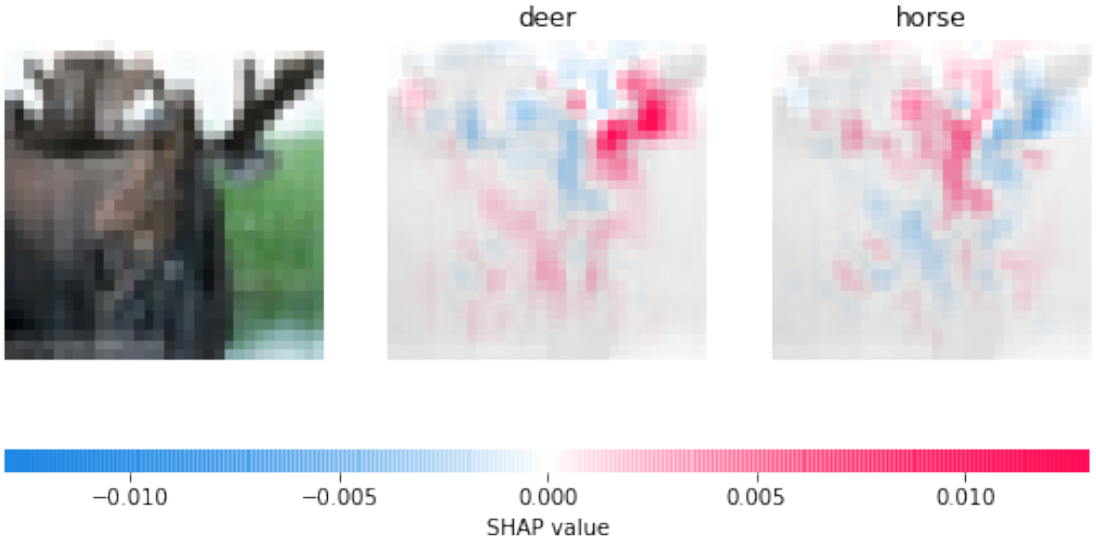


Figure 62. Kernel Shap Interpretation Sample Horse Image 4

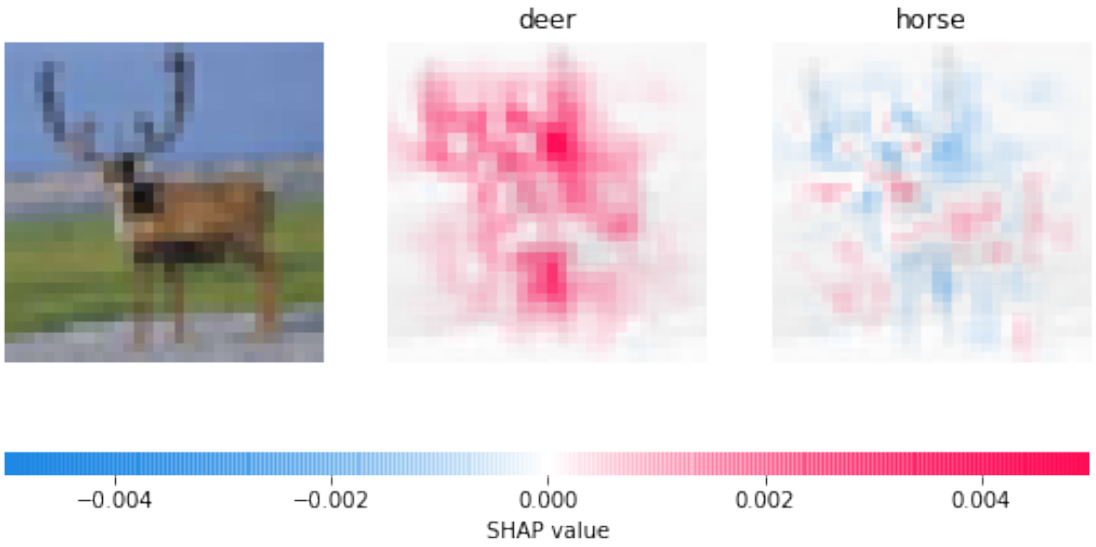


Figure 63. Kernel Shap Interpretation Sample Horse Image 5

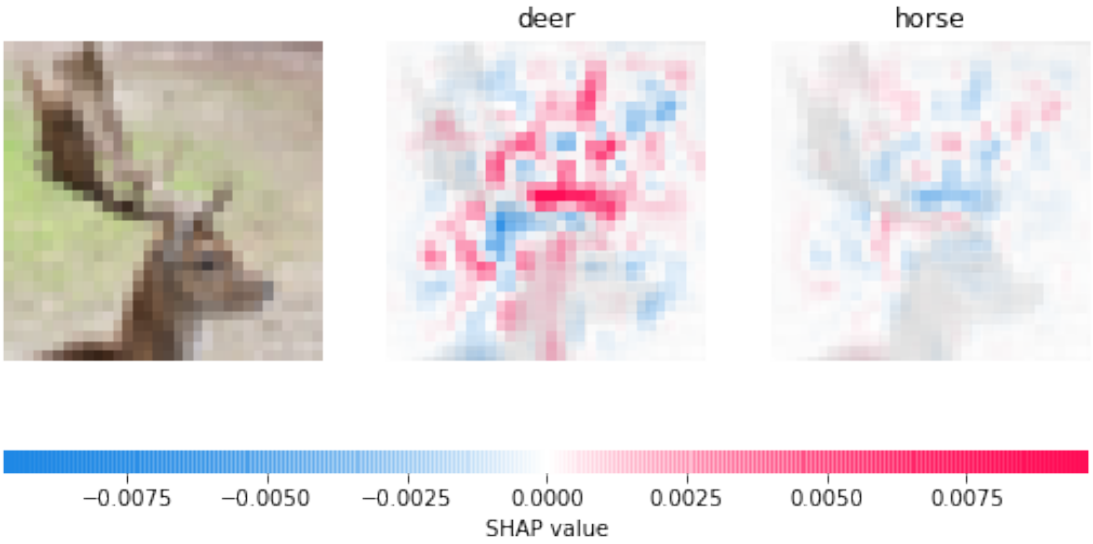


Figure 64. Kernel Shap Interpretation Sample Horse Image 6

11.6.4 Deer Interpretation Samples

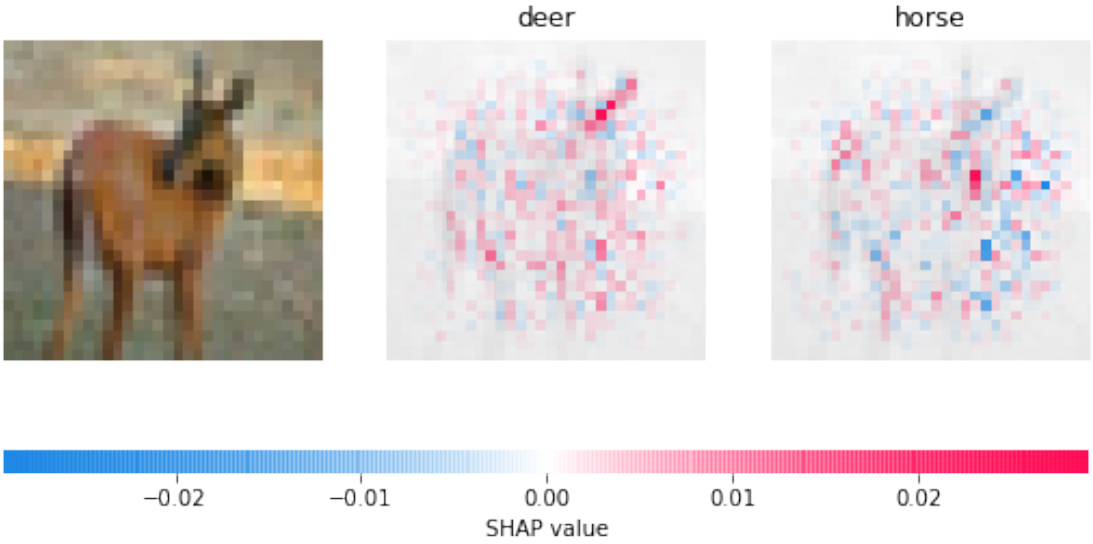


Figure 65. Kernel Shap Interpretation Sample Deer Image 1

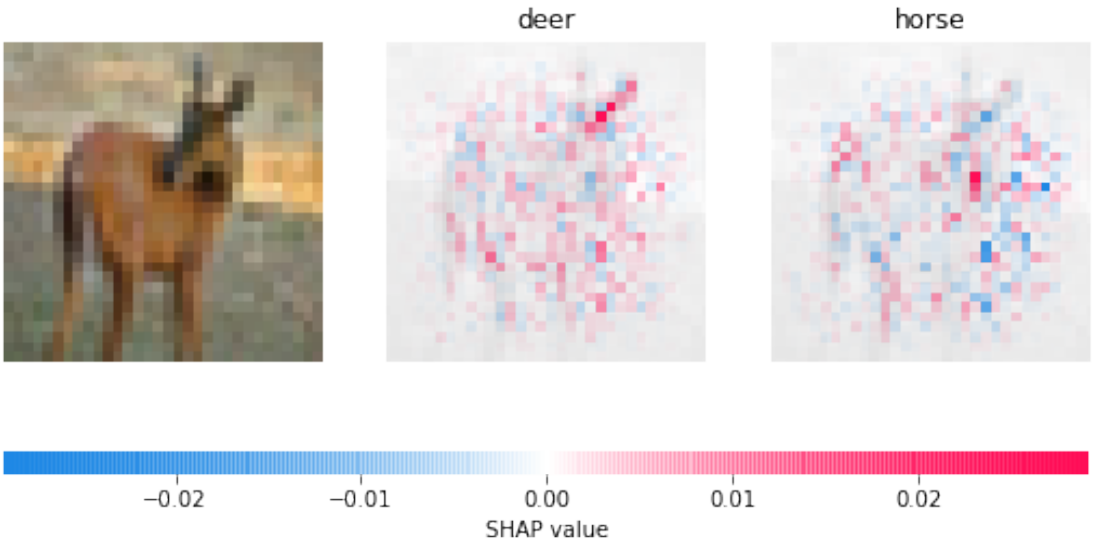


Figure 66. Kernel Shap Interpretation Sample Deer Image 1

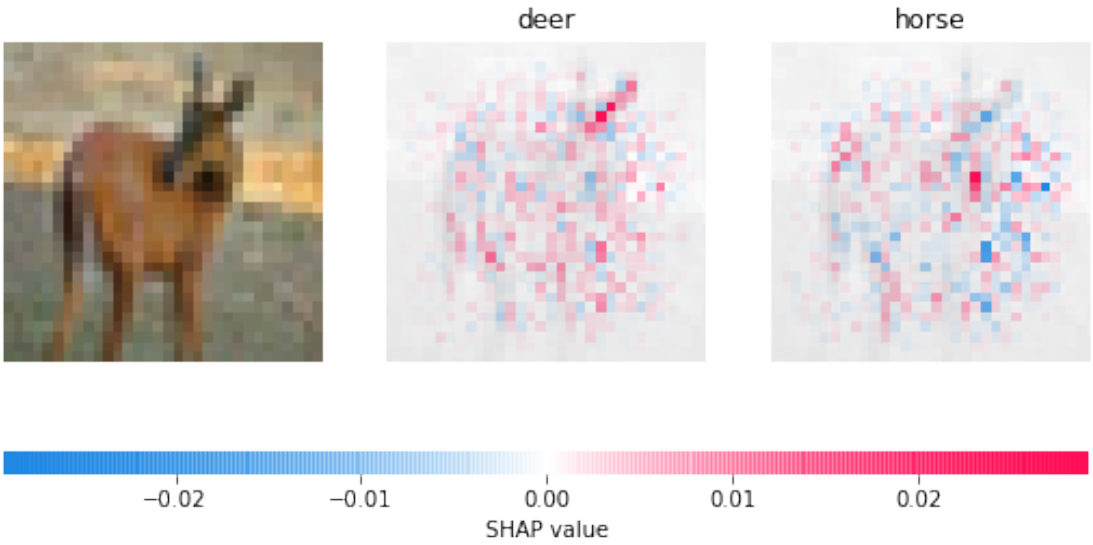


Figure 67. Kernel Shap Interpretation Sample Deer Image 2

Figure 68. Kernel Shap Interpretation Sample Deer Image 3

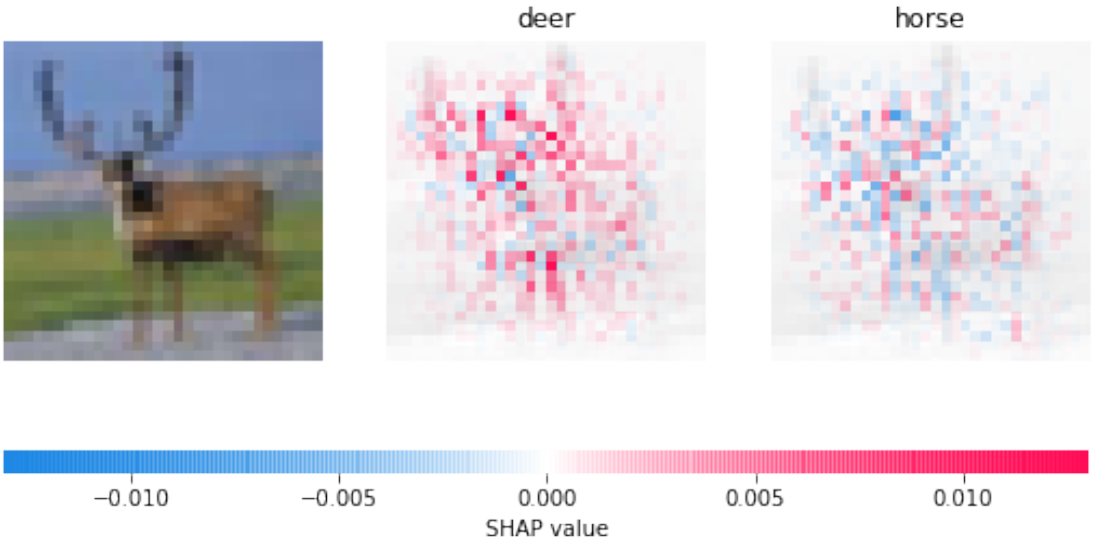


Figure 69. Kernel Shap Interpretation Sample Deer Image 4

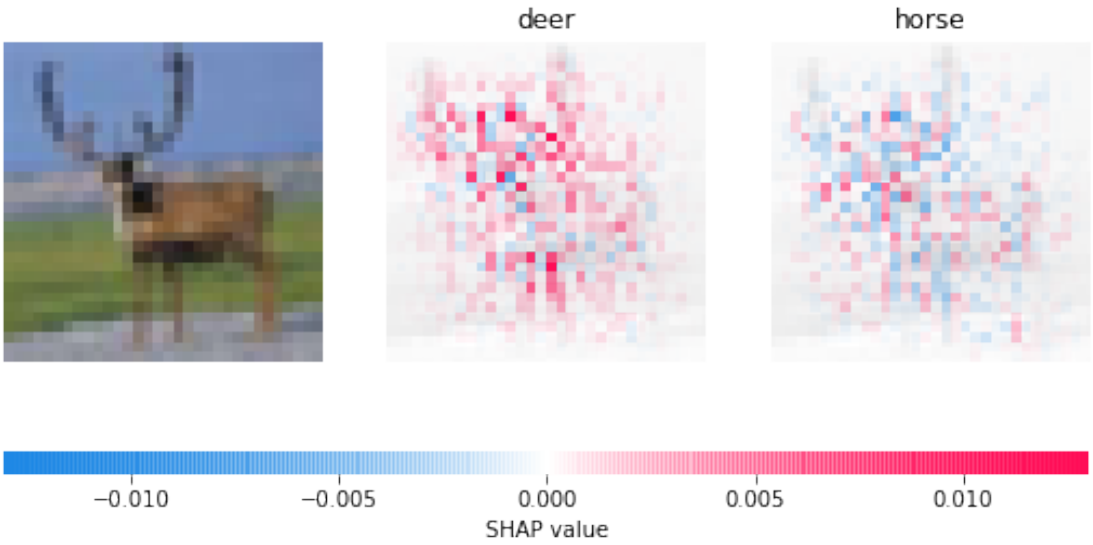


Figure 70. Kernel Shap Interpretation Sample Deer Image 5

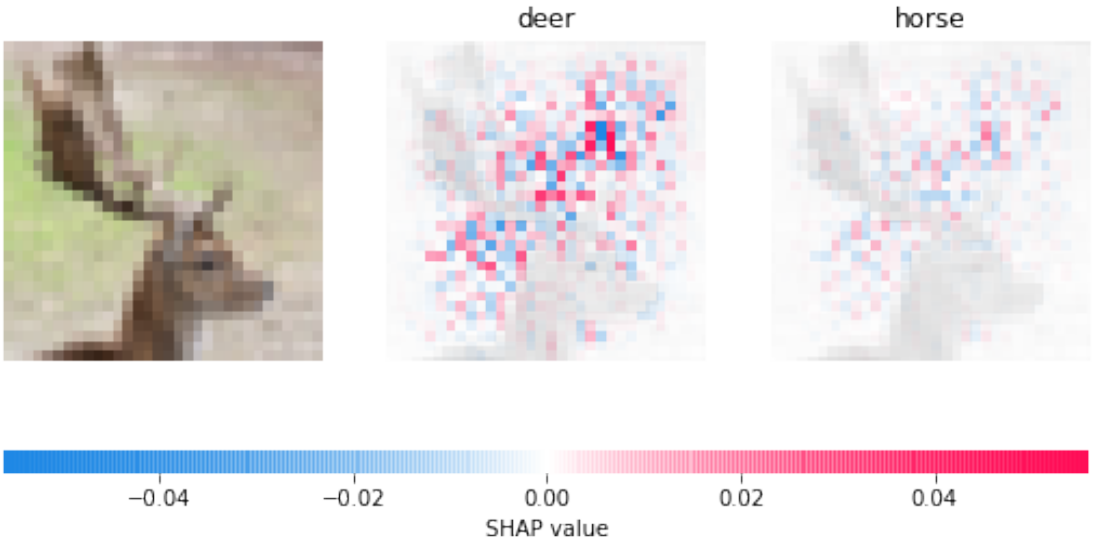


Figure 71. Kernel Shap Interpretation Sample Deer Image 6

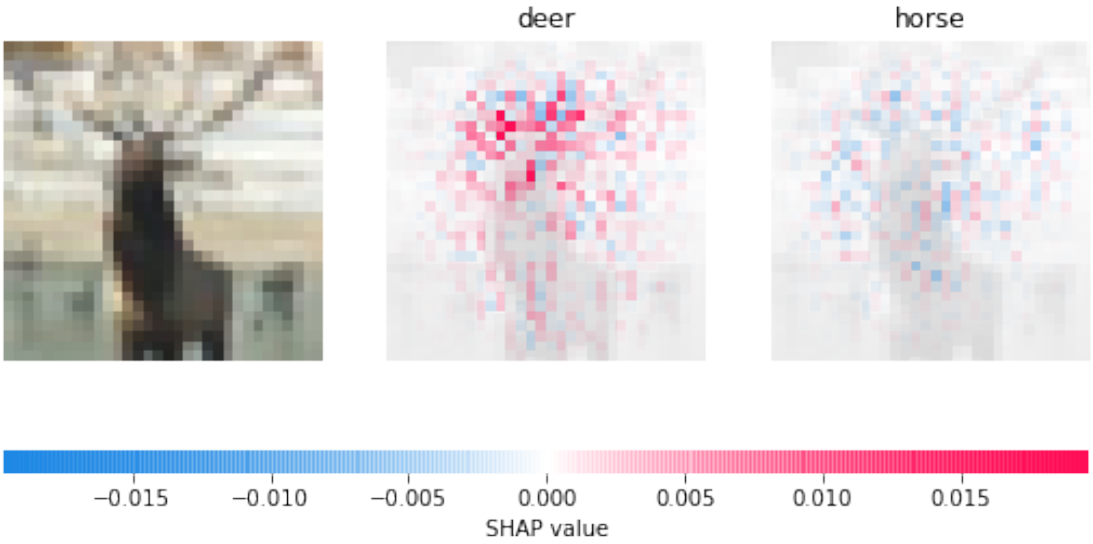


Figure 72. Kernel Shap Interpretation Sample Deer Image 7

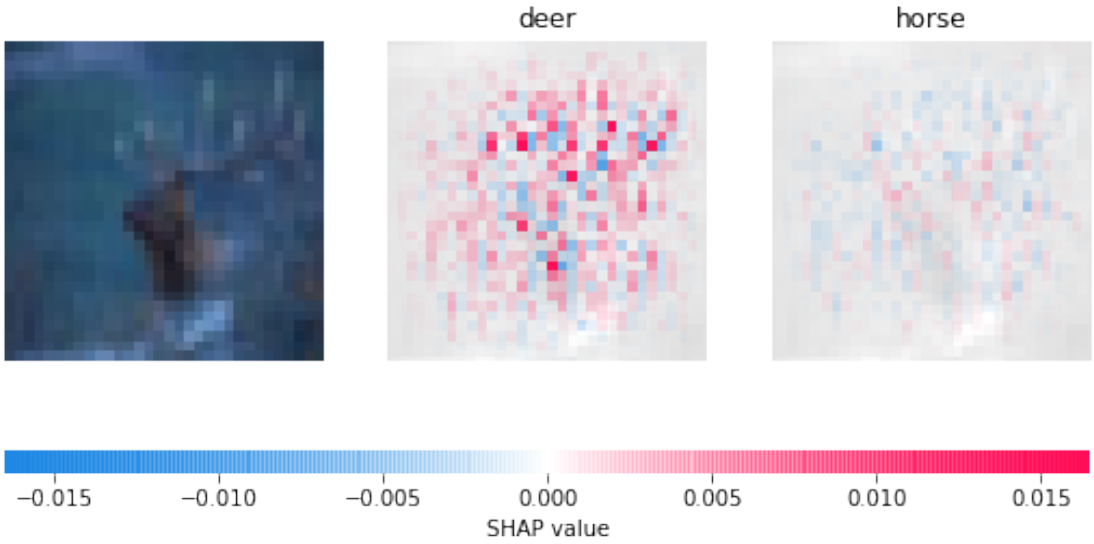


Figure 73. Kernel Shap Interpretation Sample Deer Image 8

11.6.5 Random Sampled Images

Here you can see random sampled images which explains why each image is labelled in each category.

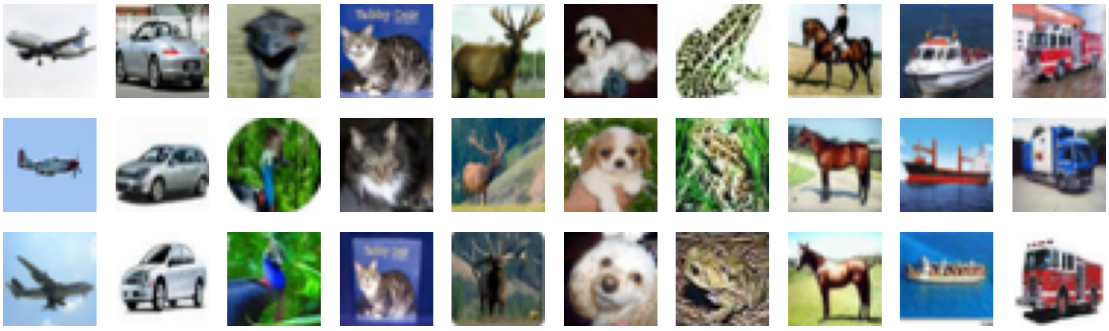


Figure 74. Kernel Shap Interpretation Sampled Images

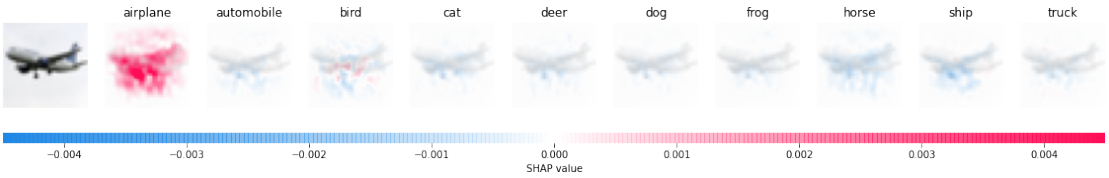


Figure 75. Kernel Shap Interpretation Sampled Image 1

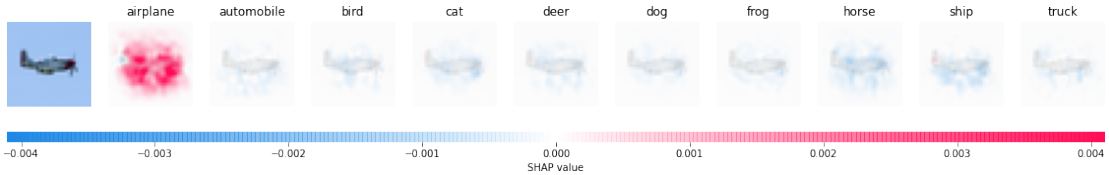


Figure 76. Kernel Shap Interpretation Sampled Image 2

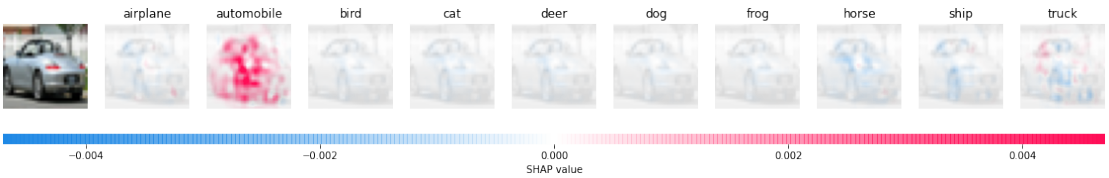


Figure 77. Kernel Shap Interpretation Sampled Image 3

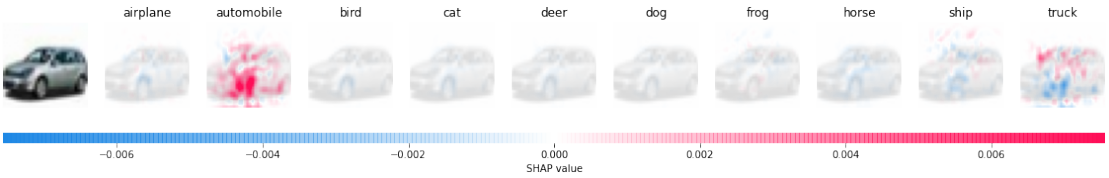


Figure 78. Kernel Shap Interpretation Sampled Image 4

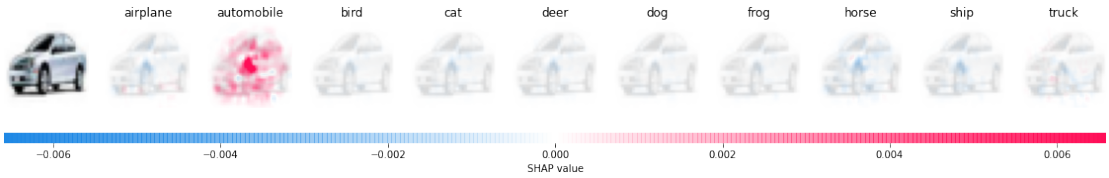


Figure 79. Kernel Shap Interpretation Sampled Image 5

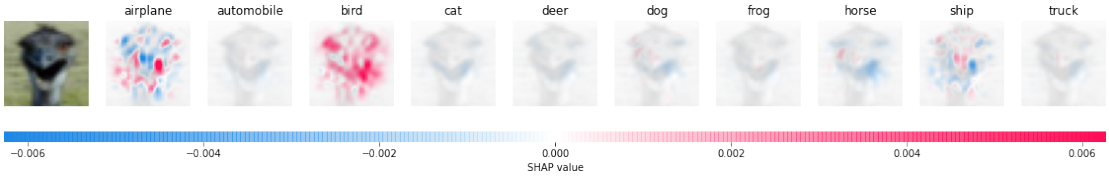


Figure 80. Kernel Shap Interpretation Sampled Image 6

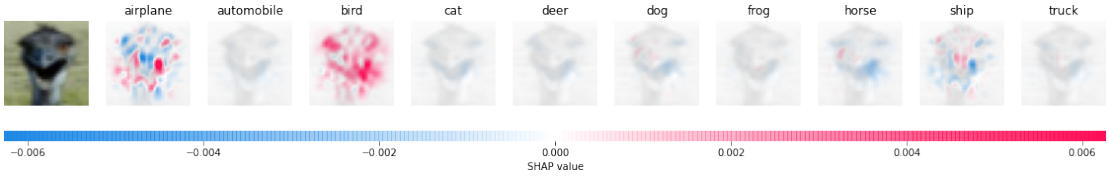


Figure 81. Kernel Shap Interpretation Sampled Image 7

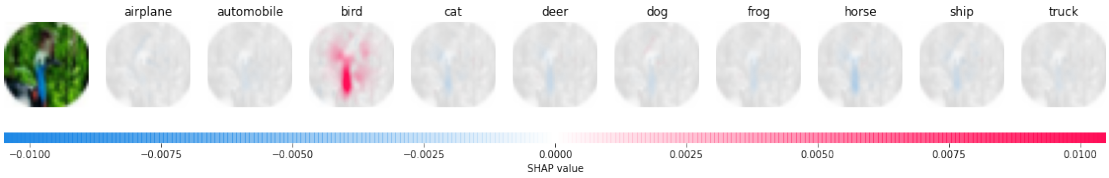


Figure 82. Kernel Shap Interpretation Sampled Image 8

Figure 83. Kernel Shap Interpretation Sampled Image 10

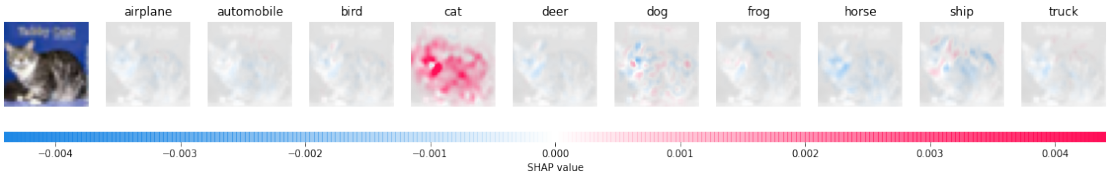


Figure 84. Kernel Shap Interpretation Sampled Image 11

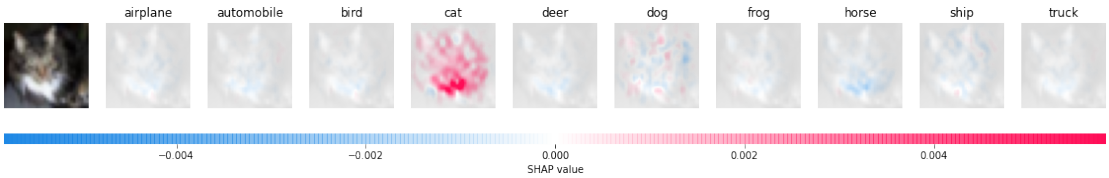


Figure 85. Kernel Shap Interpretation Sampled Image 12

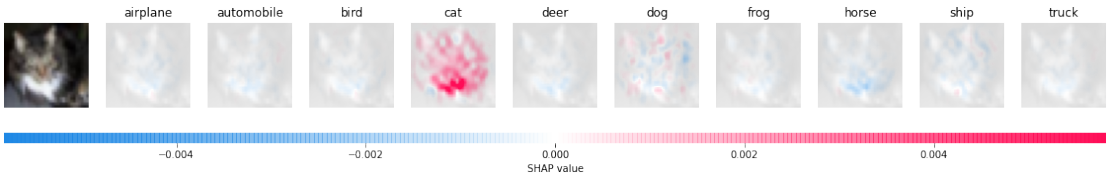


Figure 86. Kernel Shap Interpretation Sampled Image 13

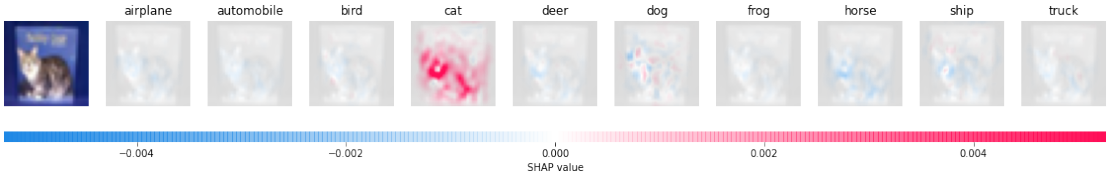


Figure 87. Kernel Shap Interpretation Sampled Image 14

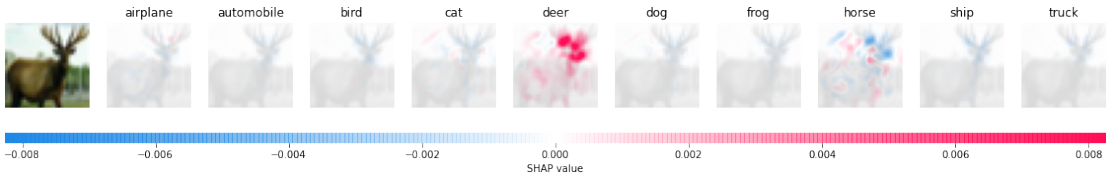


Figure 88. Kernel Shap Interpretation Sampled Image 15

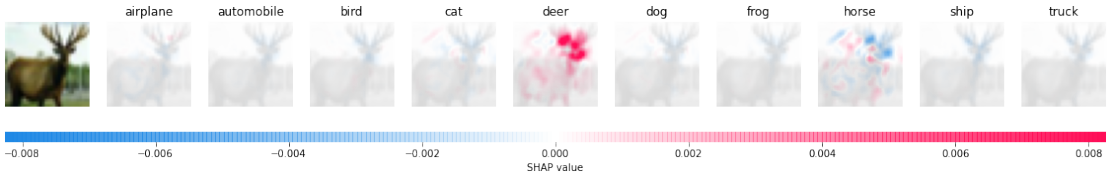


Figure 89. Kernel Shap Interpretation Sampled Image 16

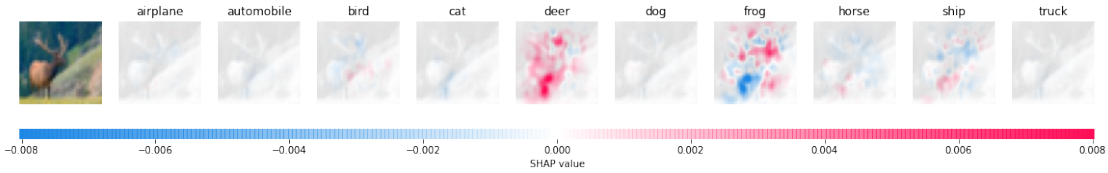


Figure 90. Kernel Shap Interpretation Sampled Image 17

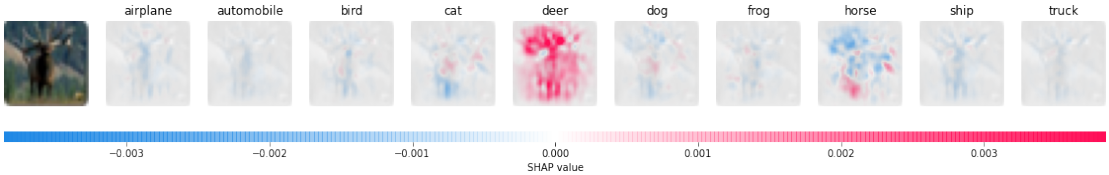


Figure 91. Kernel Shap Interpretation Sampled Image 18

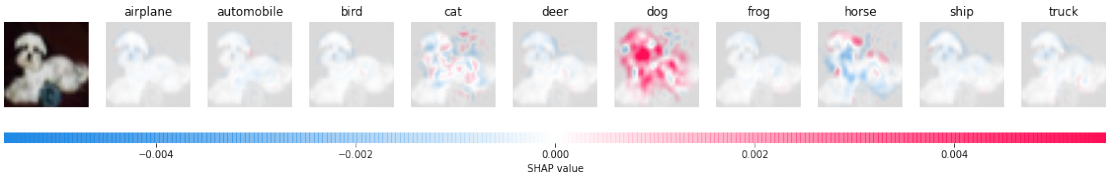


Figure 92. Kernel Shap Interpretation Sampled Image 19

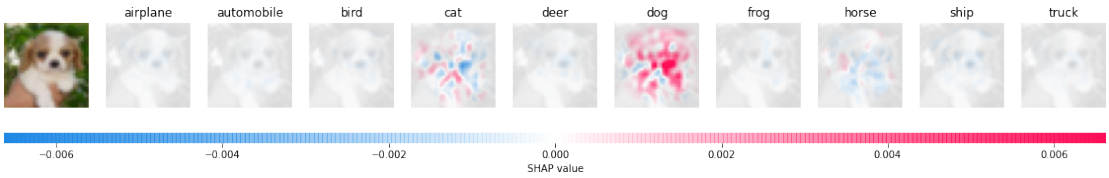


Figure 93. Kernel Shap Interpretation Sampled Image 20

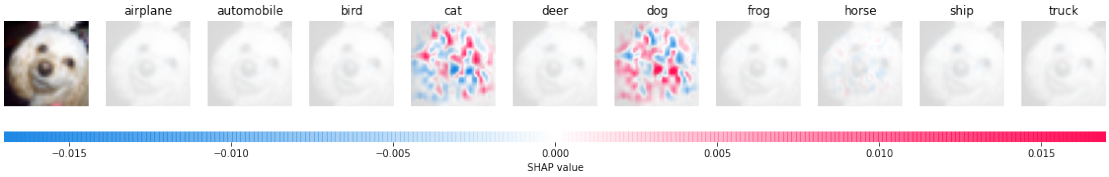


Figure 94. Kernel Shap Interpretation Sampled Image 21

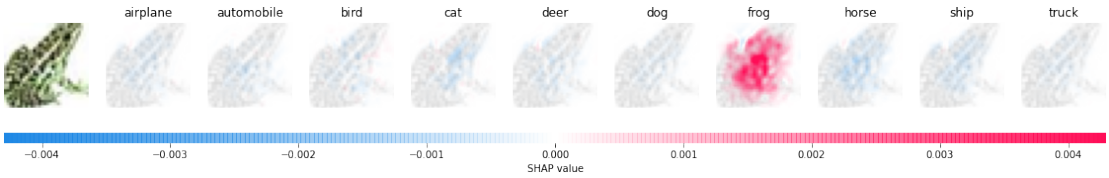


Figure 95. Kernel Shap Interpretation Sampled Image 22

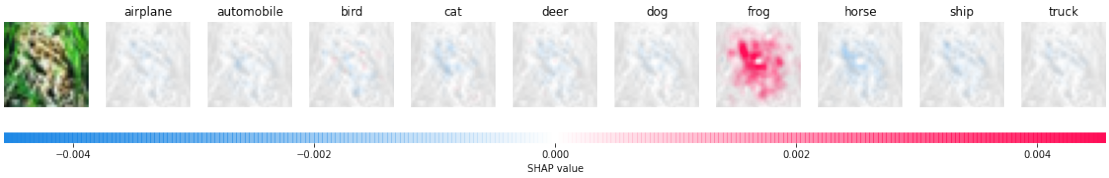


Figure 96. Kernel Shap Interpretation Sampled Image 23

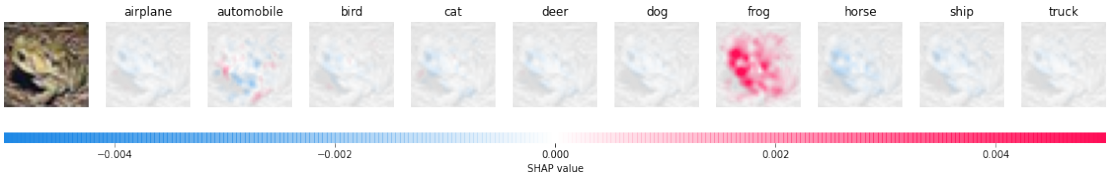


Figure 97. Kernel Shap Interpretation Sampled Image 24

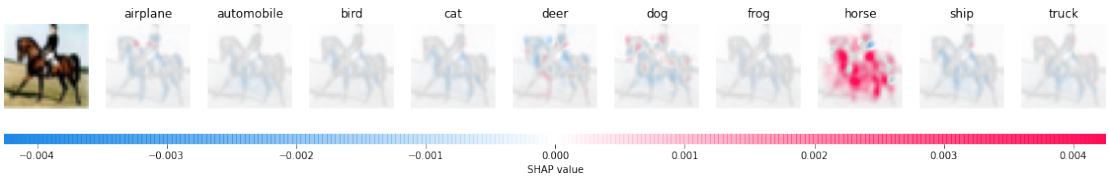


Figure 98. Kernel Shap Interpretation Sampled Image 25

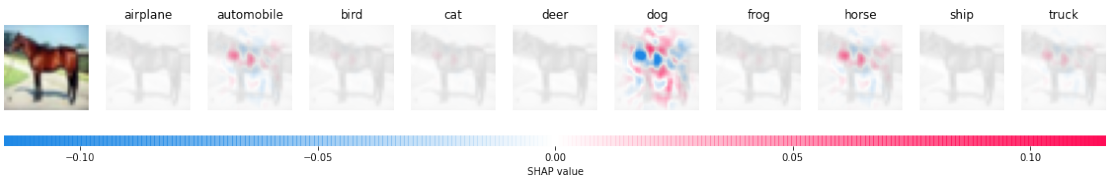


Figure 99. Kernel Shap Interpretation Sampled Image 26

11.7 MNIST KernelShap

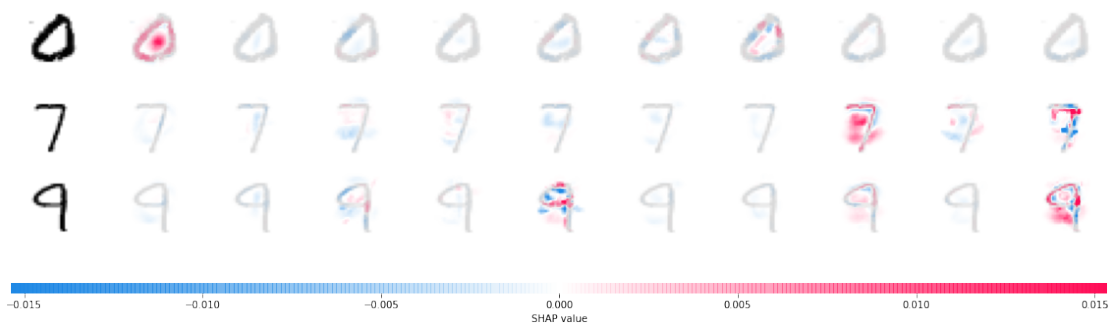


Figure 100. Kernel Shap Interpretation Sampled Image 1

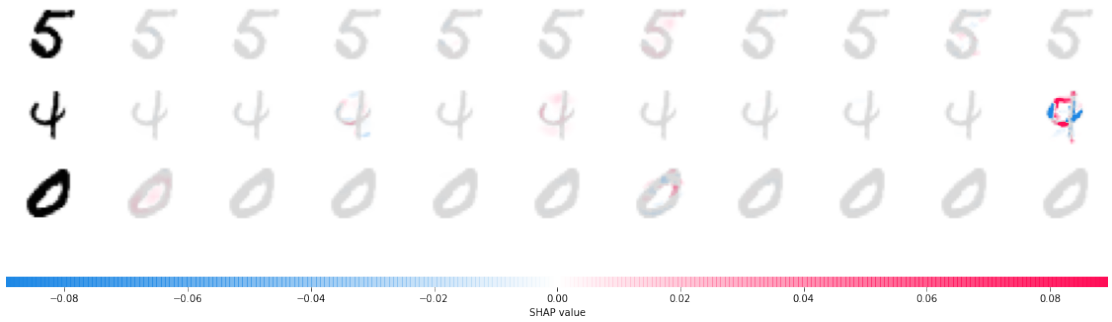


Figure 101. Kernel Shap Interpretation Sampled Image 2

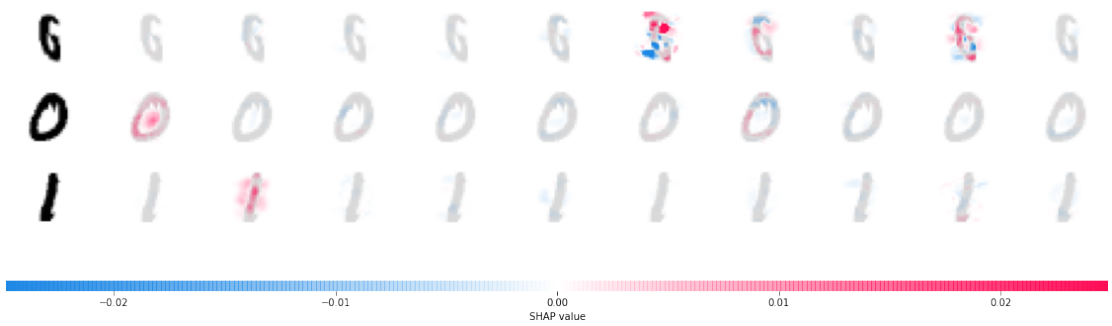


Figure 102. Kernel Shap Interpretation Sampled Image 3

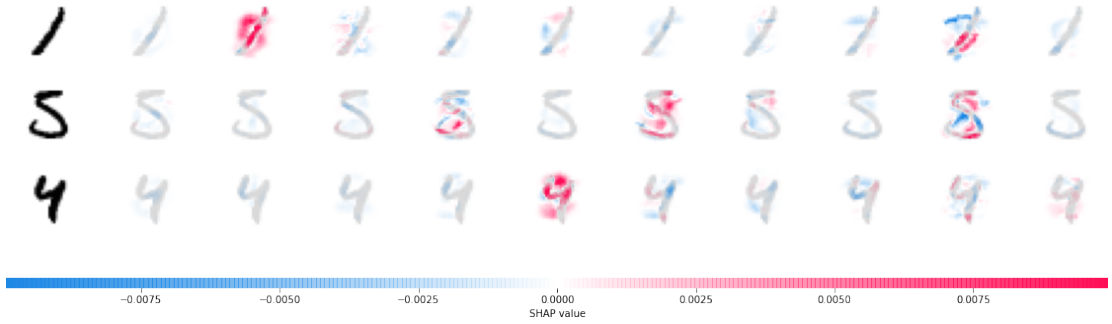


Figure 103. Kernel Shap Interpretation Sampled Image 4

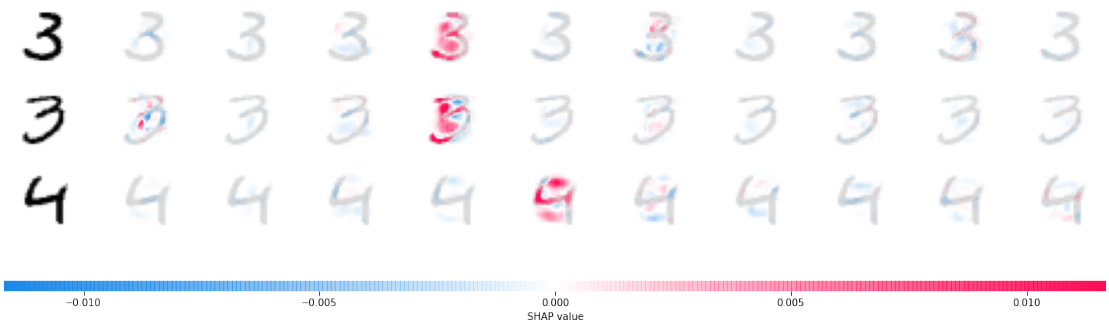


Figure 104. Kernel Shap Interpretation Sampled Image 5

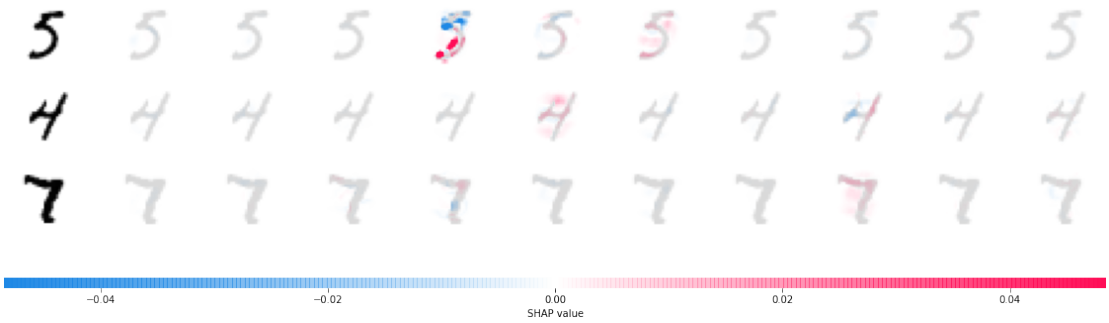


Figure 105. Kernel Shap Interpretation Sampled Image 6

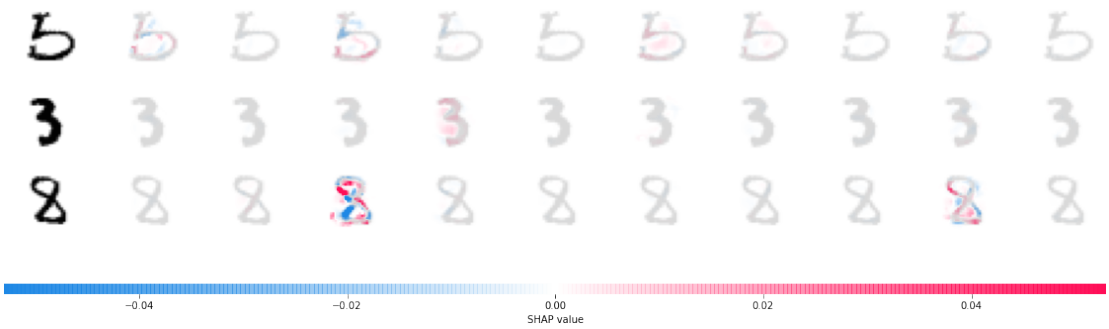


Figure 106. Kernel Shap Interpretation Sampled Image 7

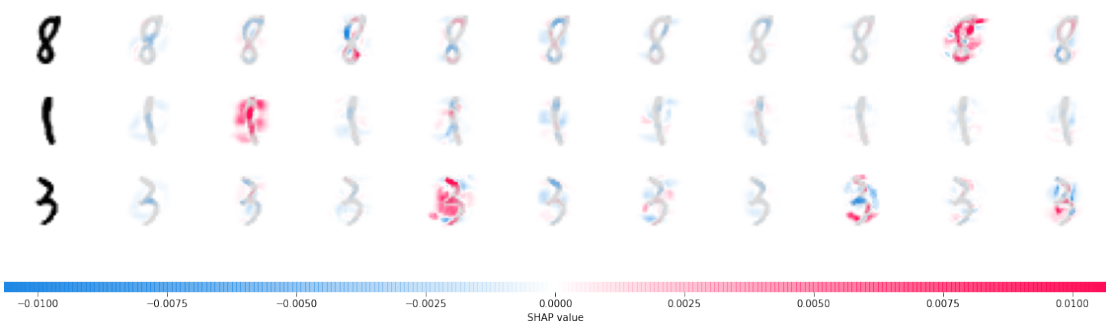


Figure 107. Kernel Shap Interpretation Sampled Image 8

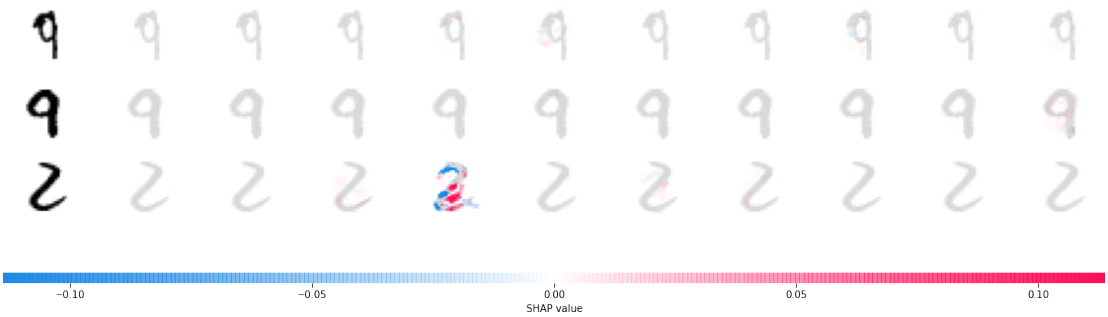


Figure 108. Kernel Shap Interpretation Sampled Image 9

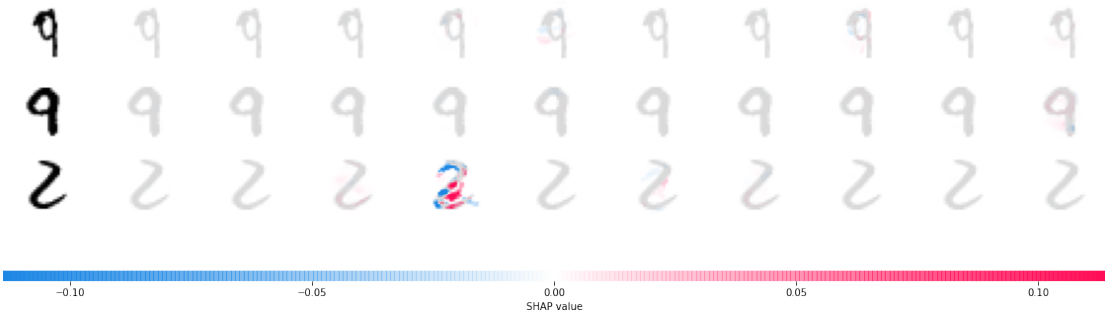


Figure 109. Kernel Shap Interpretation Sampled Image 10

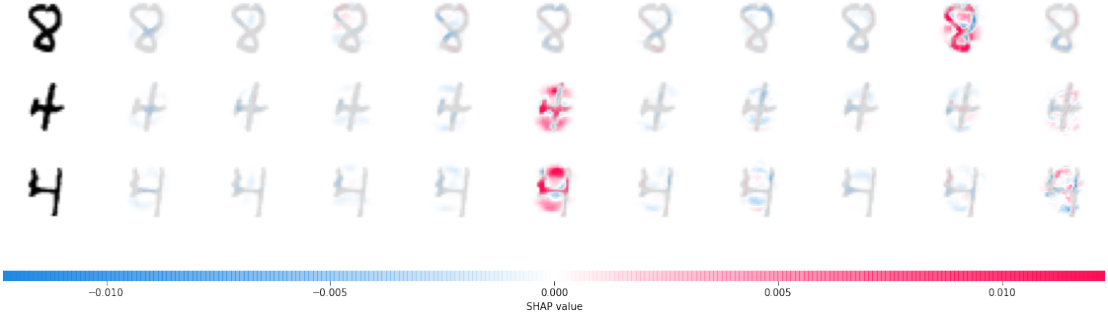


Figure 110. Kernel Shap Interpretation Sampled Image 11