CNN Model Training

Task 1:

1. Transformations:

The images are transformed as said below:

- Resize: The images are resized to a fixed size of (224, 224) pixels using the transforms.Resize transformation. This step ensures that all images have consistent dimensions, which is a requirement for many deep learning models.
- Convert to Tensor: The images are converted to PyTorch tensors using the transforms.ToTensor transformation. This step converts the images from PIL (Python Imaging Library) format to PyTorch tensors, making them compatible with PyTorch's data processing pipelines.
- Normalization: The pixel values of the images are normalized using the provided mean and standard deviation values. This normalization step helps in stabilizing the training process by bringing the pixel values within a similar range.

Since the number of images per class was imbalance, using algorithms such as SMOTE to bring up the number of samples in some classes might have been very effective in improving the models output, but after seeing that the model is already doing a decent job, I didn't look further into applying this.

We also make a validation set at this section, with the size of 0.1*train size.

2. Model Architecture

At first I decided to use AlexNet, and I implemented that in tensorflow and the results were decent and the training was faster, but coming to the second task, I had issues that I couldn't solve till the end so I switched to ResNet.

Also, after doing some research I figured that VGG is more suitable for very large datasets, so I did not consider implementing that.

Here is a brief description of ResNet:

The ResNet architecture consists of two main components: the Residual Block and the ResNet itself.

2.1 Residual Block

The ResidualBlock class defines the basic building block of the ResNet architecture. Each block is responsible for learning residual functions with respect to the input. The key components of the ResidualBlock class include:

- Two convolutional layers (conv1 and conv2) with batch normalization (bn1 and bn2) and ReLU activation.
- The forward method applies the convolutional layers to the input tensor x, computes the residual, and adds it to the output. If downsampling is required, the downsample option is applied to match dimensions.

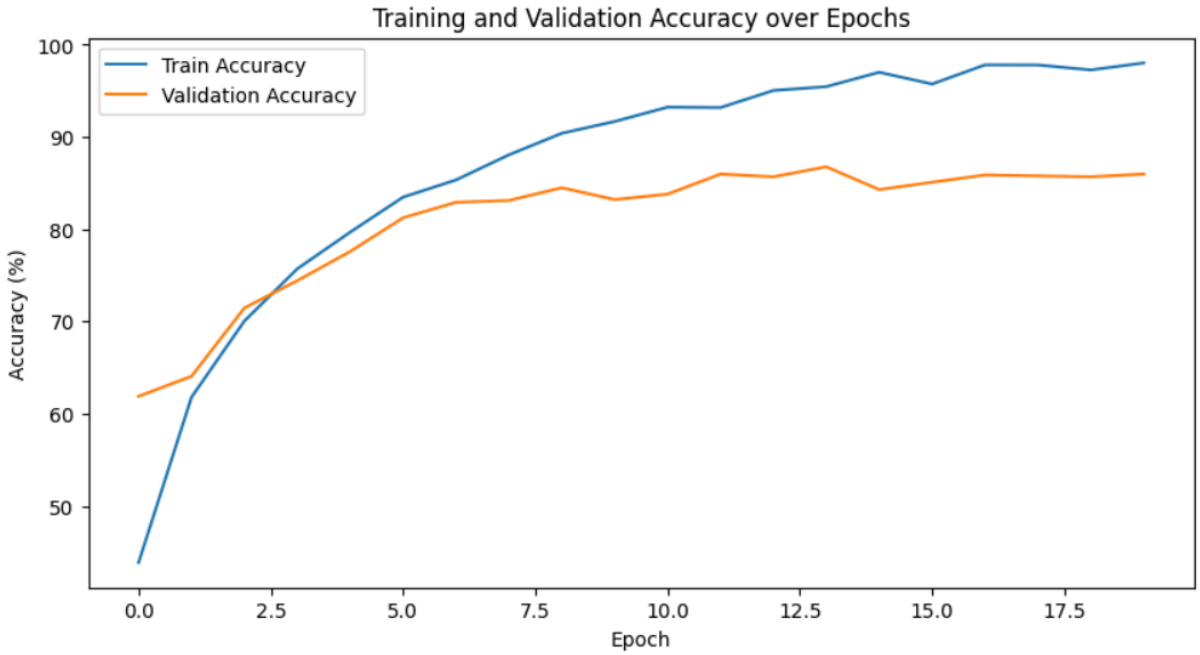
2.2 ResNet

The ResNet class implements the overall ResNet architecture by stacking multiple residual blocks. Key components of the ResNet class include:

- Initial convolutional layer (conv1) followed by batch normalization and ReLU activation.
- Max pooling layer for downsampling.
- Multiple layers (layer0, layer1, layer2, layer3) comprising stacked residual blocks with increasing feature map sizes.
- Global average pooling layer (avgpool) to reduce spatial dimensions.
- Fully connected layer (fc) for final classification.

Then, for training, the cross-entropy loss function is utilized, which is suitable for multi-class classification tasks. Stochastic Gradient Descent (SGD) optimizer is employed with a learning rate of 0.01, weight decay of 0.001, and momentum of 0.9.

The model worked pretty decent, with these being the outputs:



(No sign of overfitting)

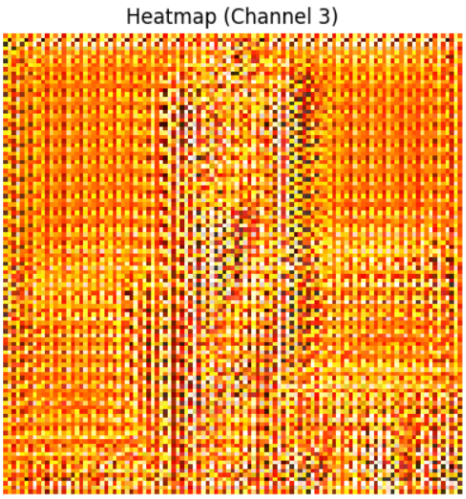
Also:
Accuracy of the network on the 1404 test images: 83.12 %
Precision: 0.8374 (meaning the model has a relatively low rate of false positives)
Recall: 0.8312 (indicating that the model has a relatively low rate of false negatives)

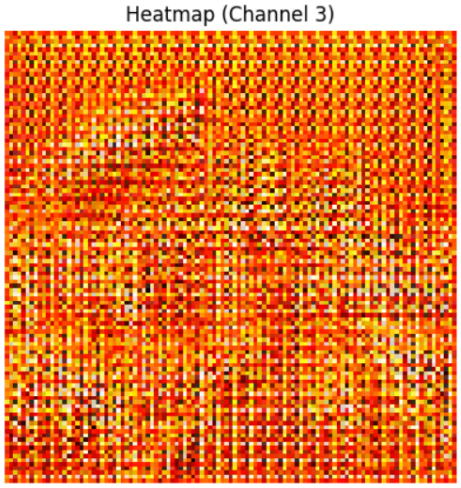
Task 2:

For this task, the `Visualizer` class defines a deconvolutional neural network (deconvnet) with two deconvolutional layers to visualize learned features from a CNN model. Each deconvolutional layer upsamples input feature maps using transpose convolution operations followed by ReLU activation. The final output layer employs a sigmoid activation to ensure the visualized feature maps are in the range [0, 1]. The `forward` method applies the sequential deconvolution layers to the input tensor to produce visualized feature maps.

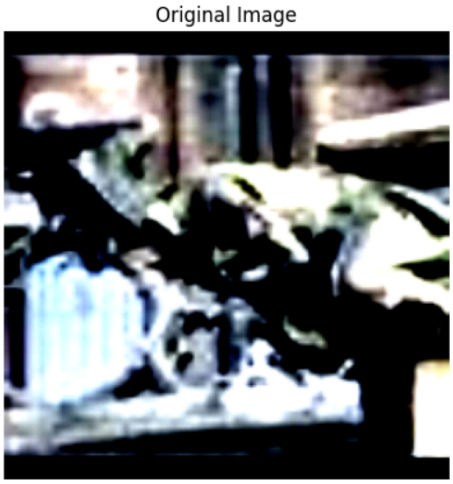
The deconvnet is used in conjunction with the main CNN model for visualization purposes. The `visualize_feature_maps` function takes the main CNN model and an input image, extracts activations from a specified layer of the model, and passes them through the deconvnet to obtain visualized feature maps. These maps are then plotted alongside the original input image for visualization and analysis using the

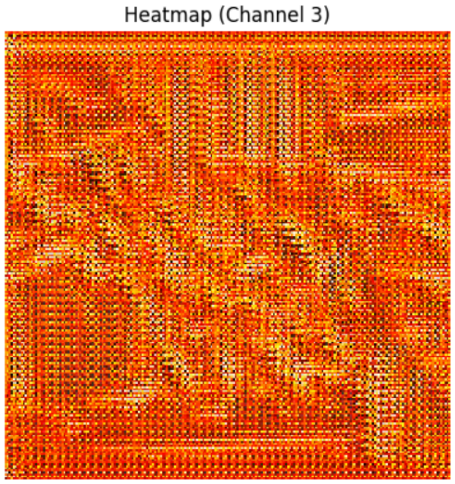
`plot_heatmap_with_image` function. This process provides insights into how the model interprets input images and identifies crucial features for classification.

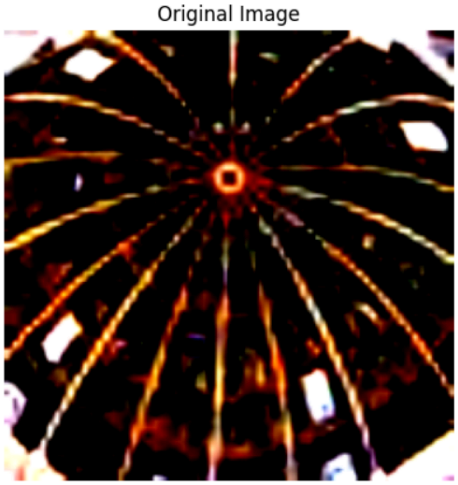


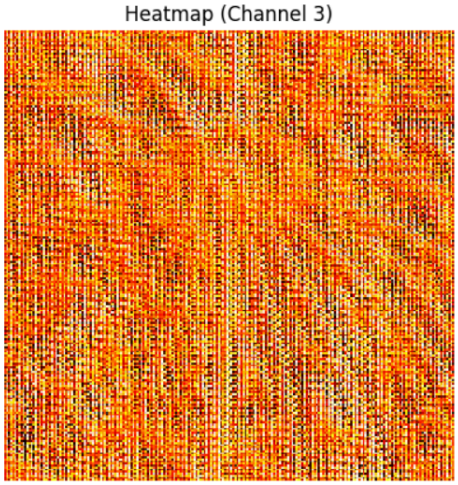


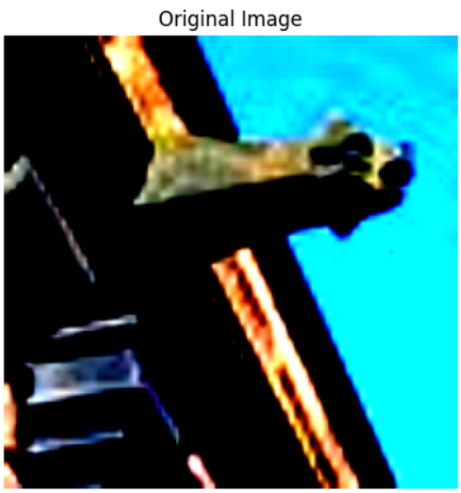
As seen here, I trained the model once more, but this time I removed the pooling layers to see how the results change.
Pooling layers reduce the spatial dimensions of the feature maps, which can lead to loss of fine-grained spatial details.
With the removal of pooling layers, the feature maps have higher spatial resolution, which can result in more detailed and informative visualizations.
These are the output for second task without having pooling layers in the model structure.:

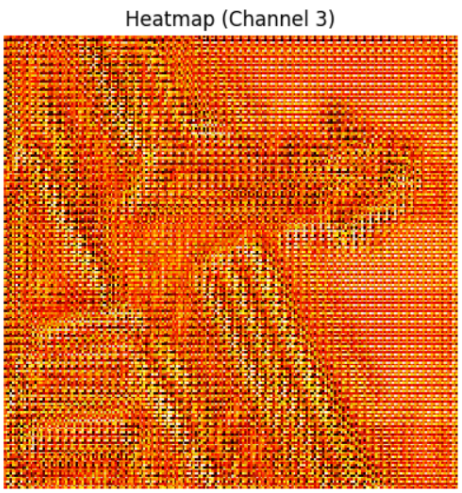


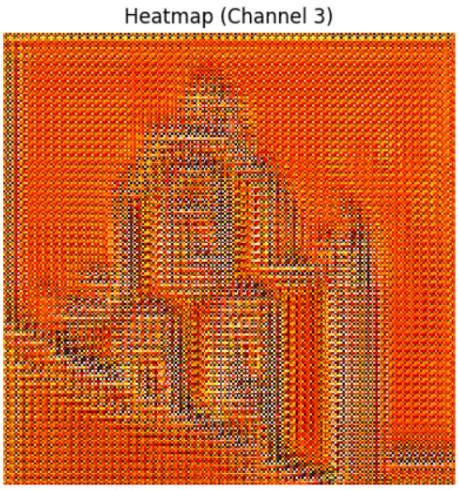










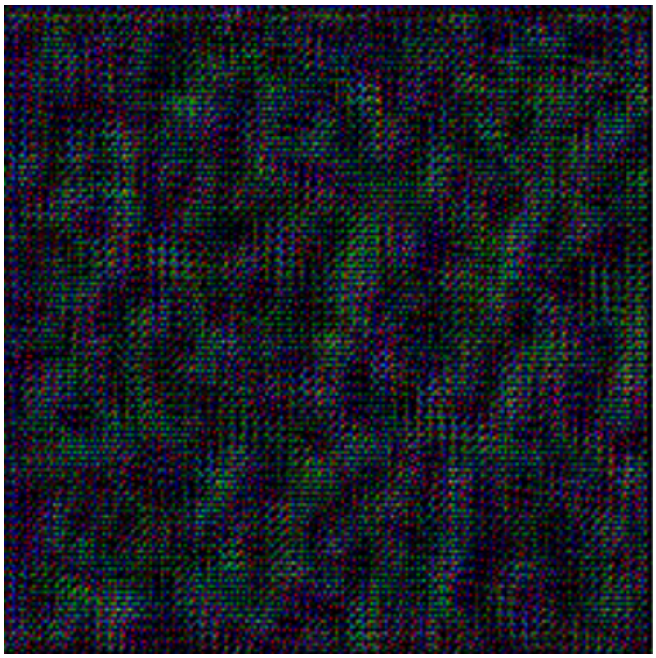


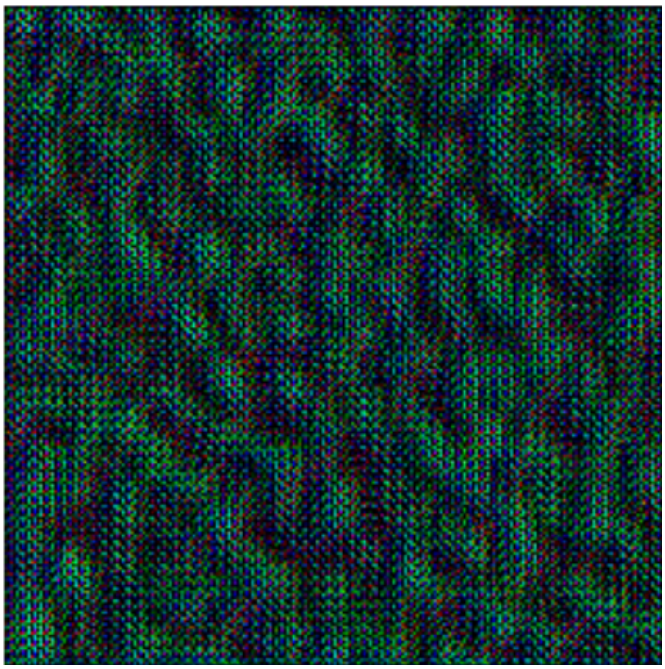
As you can see, it’s much more detailed.

Task 3:

The goal of this task is to visualize the features learned by the CNN model corresponding to a specific target class. This visualization should help us understand which features are important for the model's classification decision.
This function used both CNN and deconvnet.
The generate_image_for_class function takes parameters such as the CNN model, deconvnet visualizer, and target class name. Then a placeholder image with random noise is created as the initial input for visualization. We pass the image through the CNN model to obtain activations at specified layers.
Lastly, activations from the desired layer are passed through the deconvnet to reconstruct an image that highlights the features learned by the CNN model.

So by logic, it should give us a picture with highlight on what the model sees during training, but the pictures come out like this:





Which unfortunately has nothing to understand from.