ELEC475 Lab 4 Report

YODA: You Only Detect Anchors

November 20th, 2023

Abdellah Ghassel (20230384)

Liam Salass (20229595)

"We do hereby verify that this written lab report is our own work and contains our own original ideas, concepts, and designs. No portion of this report has been copied in whole or in part from another source, with the possible exception of properly referenced material"

Model

Modified VGG

We used a Modified VGG architecture for our YODA classifier, called Mod. This was the same model used in our previous Lab 3 CIFAR-100 classification task. The model consists of a VGG encoder pre-trained on the ImageNet dataset, which is then fed into our ModFrontend.

The ModFrontend takes a unique approach of applying an attention mechanism in the form of a Squeeze-and-Excitation (SE) block that is used to enhance the representative capacity of the network by modelling channel interdependencies. The motivation behind this was due to quick training time and improved results, and it was used by the winners of the ILSVRC 2017 Classification Competition, SENet Implementation.

The SqueezeExcitationBlock applies a *squeeze* operation via average pooling, followed by an *excitation* operation, two fully connected layers that perform recalibration through first reduction and then expansion of the dimensions. The output is passed through a sigmoid activation function.

The Mod model's frontend begins with the pre-trained encoder to extract features from the input. After encoding, an adaptive average pooling layer processes the output, which outputs a fixed-size tensor. Then, a series of layers is applied:

1. Flatten to convert the 2D tensor into a 1D tensor.
2. A Linear layer that expands the dimensions from 512 to 1024.
3. BatchNorm1d to normalize the features.
4. A ReLU activation function.
5. The SqueezeExcitationBlock recalibrates the feature channels.
6. Dropout to regularize the model by randomly zeroing 50% of the features.
7. Another Linear layer to reduce the features from 1024 to 512.
8. Another series of batch normalization and ReLU.
9. Another Dropout layer for additional regularization to prevent overfitting.
10. Final fully connected layer to output the predicted class.

During the forward pass, the encoder output is pooled and reshaped before being fed through the intermediate sequence. A residual connection using a match_dimension linear layer that maps the pooled output to the output of the intermediate sequence. This is then passed through a classifier, a linear layer that maps the learned features to the 100 CIFAR classes.

Network Diagram

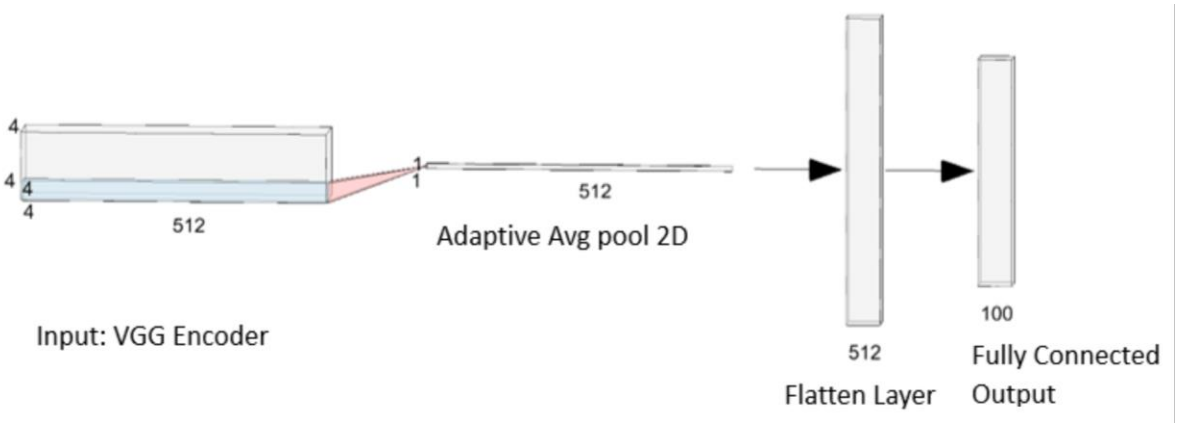

Figure 1: Netword Diagram of the Frontend Model

Model ROIs Training and Performance

Our model reached an accuracy of 95.46% on the Kitti8 ROIs generated dataset and converged after seven epochs. The training was done with the 'train_YODA_classifier.py' file. Subsequent epochs resulted in the model overfitting the training data, as seen below in the loss plot. The loss function used was the Binary Cross Entropy with Logits Loss with a positive reweighting of 3.0348. The positive weight was used to help with the unbalanced nature of the dataset, in which there are more images of non-cars than cars. This reweighting in the loss function would penalize the model more if it performed incorrect guesses on the positive labels (car in image). When training, the images underwent data augmentation and normalization. The normalization values were derived from the mean and standard deviation of the dataset, which can be calculated by running the 'YODADadaset.py' file. To further help with overfitting the model, L2 Regularization was used to ensure that model weights would become too large and avoid vanishing gradients, leading to overfitting.

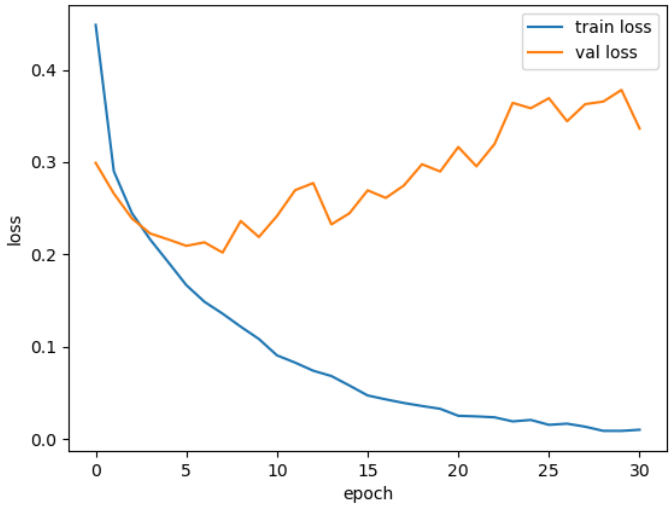

Figure 2: Loss Plot of Modified VGG Encoder

The following are the performance metrics of our model on the Kitti8 ROIs test dataset.

Accuracy: 0.954577

F1 Score: 0.911146

Confusion Matrix:

	0 (Predicted Negative)	1 (Predicted Positive)
0 (Actual Negative)	51237	2256
1 (Actual Positive)	973	16622

Classification Report:

	Precision	Recall	F1-Score	Support
0	0.98	0.96	0.97	53493
1	0.88	0.94	0.91	17595
Accuracy			0.95	71088
Macro Average	0.93	0.95	0.94	71088
Weighted Average	0.96	0.95	0.96	71088

Model Performance

Using the test_YODA.py file, the mean IoU score of the model and images with the predicted car regions were generated and tested. The model returned a mean **IoU score of 0.11243**, which is relatively low. Ideally, an IoU score of 0.5 is acceptable for object detection tasks.

Table 1: Example output predictions with generated prediction boxes and IOU score overlayed.

Discussion

In this discussion section, we will assess the performance of our YODA (You Only Detect Anchors) system, evaluate whether it met our expectations, and delve into the challenges we encountered during its development and deployment.

While our YODA system performed well in terms of classification accuracy, we encountered some challenges during its development:

1. Overfitting: One of the challenges we faced was the model's tendency to overfit the training data, as indicated by the loss plot. To mitigate this, we employed dropout layers and

regularization techniques to prevent the model from memorizing the training data and improve its generalization.

2. Low Mean IoU Score: Another significant challenge was the relatively low mean Intersection over Union (IoU) score of 0.11243 obtained during object detection testing. An IoU score of 0.5 is typically considered acceptable for object detection tasks, indicating that our model struggled with accurately localizing objects within the images. To address this, further research and experimentation with different object detection techniques and architectures may be necessary to improve localization accuracy.
3. Class Imbalance: The dataset used for training contained more images of non-cars than cars, leading to a class imbalance issue. We addressed this by applying a positive reweighting factor in the loss function. However, further exploration of techniques like data augmentation or collecting a more balanced dataset could help improve model performance in object detection.

In summary, our YODA system demonstrated promising results in terms of classification accuracy, but there is room for improvement in object detection accuracy. The challenges we encountered, such as overfitting and low IoU scores, point to areas where optimization efforts can be focused to enhance the system's overall performance.