Analyzing Machine Learning Models for ADE20K Dataset

Apetrii Radu-Andrei
Artificial Intelligence and Optimization Masters

Baranceanu Vlad-Andrei
Artificial Intelligence and Optimization Masters

***Abstract*—This document delves into diverse endeavors centered around images sourced from the ADE20K Dataset, focusing on the segmentation of these images into distinct classes through the application of advanced Machine Learning models.**

Throughout the experiment, we have used the following models: UNet, Attention UNet, DeepLabV3 - with and without pre-trained weights -. The results are relatively poor due to the memory limitation of the environment we used. Thus, our main focus shifted towards the analytical side - comparing the models and improving metrics -.

***Index Terms*—Machine Learning, ADE20K Dataset, UNet, Attention UNet, DeepLabV3, Semantic Segmentation**

I. Introduction

A. ADE20K Standard Competition

The ADE20K Dataset encompasses a vast and diverse collection of scene-centric images. With over 20,000 annotated images across various indoor and outdoor scenes, the challenge resides in understanding the complexity of pixel-level segmentation to accurately label and classify objects within the images.

The competition is based on Kaggle, an online platform dedicated to data scientists, machine learning practitioners, and researchers that seek knowledge-extending opportunities.

The Dataset contains 150 classes which range from walls, buildings, skies, floors to waterfalls, showers, clocks and drinking glasses. These objects are not uniformly distributed across the images, some of them having a higher presence count. These images have been taken across different environments, some of them being indoors, such as in bathrooms, airport terminals, lab classrooms, study halls, while others being part of outdoor scenes, for example lagoons, snowfields, mountains. An example of an image that is part of the dataset is:

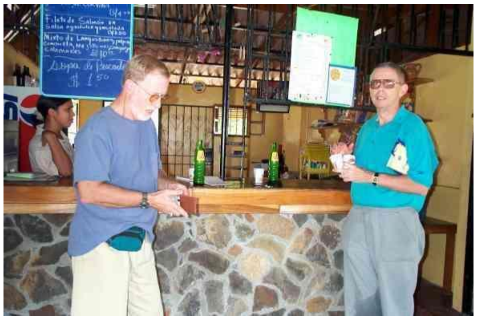


Fig. 1. Original Image

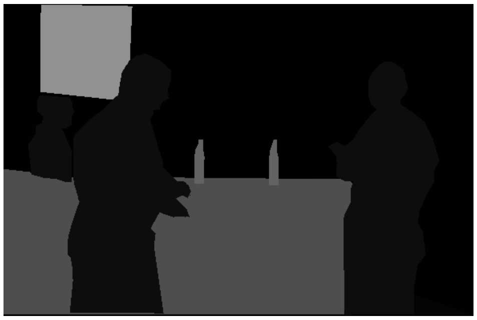


Fig. 2. Mask Image

B. Best Score Solution and Results

In order to check the accuracy of a solution, the ”Dice Score” method is applied. This is a metric commonly used to measure the similarity or overlap between two sets. In the context of image segmentation or binary classification tasks, it assesses the agreement between predicted and ground truth masks. This method uses the values of four computed metrics: True Positive, True Negative, False Positive, False Negative, by using the following formula:

$$\text{Dice Score} = \frac{2\times\text{TP}}{2\times\text{TP}+\text{FP}+\text{FN}}$$

Our best score - 1.00579 - was obtained by using Attention UNet with dropout, by training the model for 50 epochs, and by trying to predict 10 classes out of the total of 150. We have also used the Adam optimizer with a learning rate of *1e-3*. In order to compute the loss, we have opted for the Cross-Entropy function.

II. State of The Art

A. Related Articles

In terms of articles/other works that we have followed, notable mentions can be given to:

- Semantic Segmentation Using DeeplabV3 [4] - A DeepLabV3 model that tries to do semantic segmentation on self driving cars images. This is similar to our task due to the presence of outdoors images.
- Project-MONAI tutorials [5] - A GitHub repository from which we found various models for semantic segmentation task.
- milesial Pytorch-UNet - A GitHub repository which aided us into the process of designing a UNet model.

B. Other Results

There have been a lot of attempts over the years at trying to correctly predict the results from the ADE20K Dataset. Some of the best results can be found in the following graph:

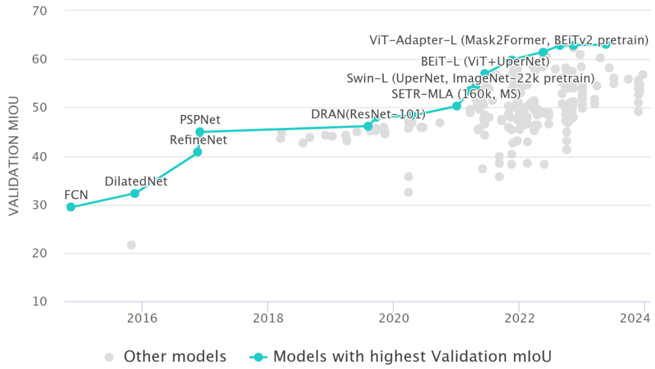


Fig. 3. Benchmark Chart

Starting from the year 2014 with a result of 29.39 - Fully Convolutional Networks -, the scores have been improving over the years, the last attempt being in 2023 with a score of 63 - ONE-PEACE -.

Comparative to the SOTA, our score is quite poor. The reasons for that include: limited time and memory resources, lots of different images from several environments which need a large model that is able to comprehend the dataset particularities.

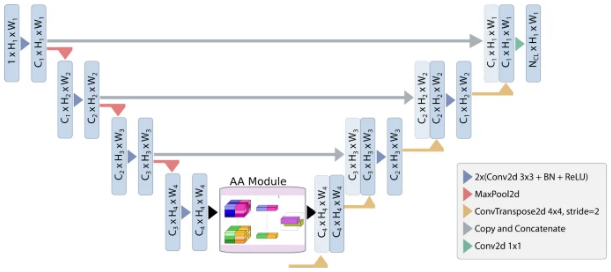
III. Our Approach

A. Benchmark Performance

The train dataset we have used contains a number of 20.000 images. For each image, we have generated a mask. The validation dataset contains the other 2.000 images. In our implementation, we have constructed a class that loads the data and applies the following transformations to the images: resize and normalization - converting the input values to [0,1] -. The resize of images is required because they have different heights and widths, and the model needs for all the images to have the same size. We have chosen a standard size of 512x512 pixels. We were not able to upload the images at a full resolution because the memory resources are limited. Also, we wish for the least amount of information lost as possible. Hence, this shape acts as a middle ground. The same transformations are applied to the masks. In addition, we turn the masks into grayscale ones because we require only one channel of color when computing the loss.

In order to upload the data, we use a data loader. The batch size depends on the model which is used on the training part, having a value of 32. Before the model embarks on its journey, the data is shuffled.

The model that lead us to the benchmark score is Attention UNet. The Attention U-Net is an advanced variant of the traditional U-Net architecture, incorporating attention mechanisms to enhance image segmentation tasks. By selectively emphasizing informative spatial and channel-wise features, this model demonstrates improved accuracy in capturing intricate details and fine-grained structures. Particularly effective in applications such as medical image segmentation and satellite image analysis, the Attention U-Net offers a refined approach to handling complex scenes and diverse objects within images, making it a valuable tool in computer vision. The model was trained for 50 epochs.



$1 \times H_1 \times W_1$
$C_1 \times H_1 \times W_1$
$C_1 \times H_2 \times W_2$
$C_2 \times H_2 \times W_2$
$C_2 \times H_3 \times W_3$
$C_3 \times H_3 \times W_3$
$C_3 \times H_4 \times W_4$
$C_4 \times H_4 \times W_4$
AA Module
$C_4 \times H_4 \times W_4$
$C_4 \times H_4 \times W_4$
$C_3 \times H_3 \times W_3$
$C_3 \times H_3 \times W_3$
$C_2 \times H_3 \times W_3$
$C_2 \times H_2 \times W_2$
$C_2 \times H_2 \times W_2$
$C_1 \times H_2 \times W_2$
$C_1 \times H_1 \times W_1$
$C_1 \times H_1 \times W_1$
$N_{CL} \times H_1 \times W_1$
2x(Conv2d 3x3 + BN + ReLU)
MaxPool2d
ConvTranspose2d 4x4, stride=2
Copy and Concatenate
Conv2d 1x1

Fig. 4. Attention UNet

Our approach consists of making predictions only for the most common 11 classes. The combined ratio of these adds up to a value of 0.6087. This decision was based on the memory limitations we were facing. Also, the learning process would have been slowed down by classes that rarely appear in the data. Thus, we have substituted their values with the most common dataset value. However, we have managed to speed up the process by using parallelization with the use of the two GPUs available.

Adam was selected as an optimizer. It seamlessly integrates momentum and adaptive learning rates, enabling faster convergence and improved efficiency during the optimization process. By dynamically adjusting learning rates for each parameter based on historical gradients, Adam excels in handling sparse gradients and non-stationary objectives. This versatility makes it widely utilized in training deep neural networks for tasks like image recognition and natural language processing.

For computing the loss we have opted for the Cross Entropy function. It quantifies the dissimilarity between predicted probability distributions and actual distributions. The formula for the Cross Entropy used in our approach is:

$$\mathrm{H(y,\ p)} = -\sum_i y_i \cdot \log(p_i)$$

The next step in our process was to run the model on the validation data. For this, we have loaded the dataset and ran the model on every image. In order to reduce the output size, we have compressed the matching consecutive results into a pair - the first number representing the class obtained as a result and the second number consisting of the number of consecutive times it appears in the output -. The score obtained by running this model was: 1.00579.

B. Ablation Study

There were several other attempts tried for the segmentation task.

1) UNet: U-Net is a convolutional neural network architecture designed for semantic image segmentation tasks. Its distinctive U-shaped structure features a contracting path to capture context and a symmetric expanding path for precise localization. U-Net has skip connections that help combine low-level and high-level features, making it highly effective for tasks like medical image segmentation and object detection, where precise delineation of objects in images is crucial.

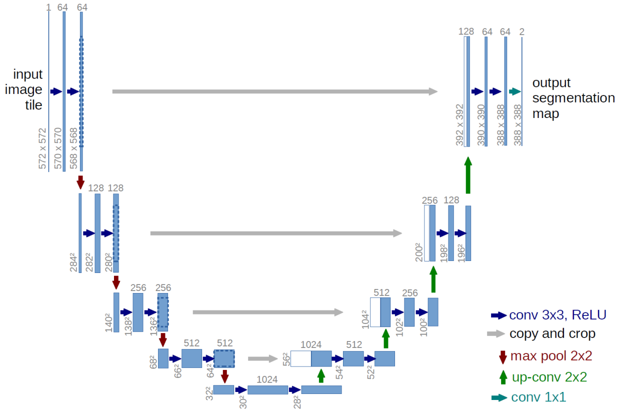


Fig. 5. Attention UNet

The first attempt was to run the UNet model on all 150 classes which was trained for 10 epochs with a batch size of 8. This produced the following result: 0.01236. The two 16GB GPUs were used up to their full potential.

Another attempt was to try running the model on 10 classes with a batch size of 16 for 50 epochs, which produced a result of: 0.73861. Same as before, the two GPUs have been used to full capacity.

2) DeepLabV3: DeepLabv3 is a state-of-the-art convolutional neural network architecture designed for semantic image segmentation. It builds upon the DeepLab family, incorporating dilated convolutions, atrous spatial pyramid pooling (ASPP), and encoder-decoder structures. DeepLabv3 excels in capturing fine details and context in images, making it particularly effective for tasks like object recognition, scene parsing, and medical image segmentation. Its ability to handle various object scales and produce detailed segmentation masks has contributed to its widespread use in computer vision applications.

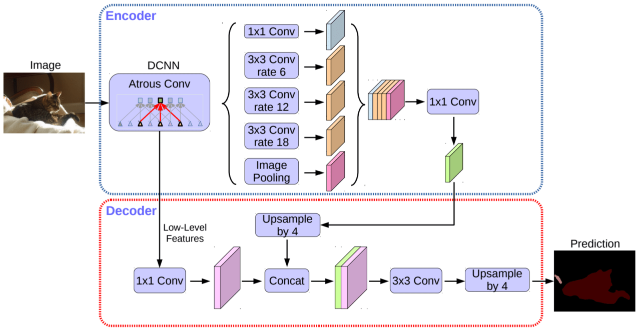


Fig. 6. Deep Lab V3

The first attempt was to run a DeepLabV3 model without pre-train on 10 epochs with a batch size of 10. This produced really poor results as it was unable to learn anything. However, the two GPUs were again used to full capacity. We were not discouraged by this attempt and we moved on to the second one.

As a second experiment we have opted for a pre-trained model on 50 epochs with a batch size of 20. In terms of results, we got the following score: 0.53589. As before, we have used the two GPUs to their maximum capacities.

3) Attention UNet: There were other attempts we have tried with Attention UNet. Mainly, we tried to continue the learning process for the benchmark model. After 125 epochs, we got the following result: 0.74429. After the model reached 200 epochs, the score that we have obtained is: 0.5686. As it can be observed from the scores, the model started to overfit.

4) Comparison: The following table presents a comparison between all models we have used throughout our experiments.

TABLE I
MODEL RESULTS

Model	Epochs	Batch size	Classes	Memory	Time	Score
UNet	10	8	150	32 GB	8 hr	0.01236
UNet	50	16	10	32 GB	36 hr	0.73861
Attention UNet	50	32	10	28 GB	8 hr	1.00579
Attention UNet	125	32	10	28 GB	20 hr	0.74429
Attention UNet	200	32	10	28 GB	31 hr	0.5686
DeepLabV3	50	20	10	32 GB	36 hr	0.53589

Based on the above table, we have extracted the following observations:

- Attention UNet needs less memory to work with. In consequence, it can use a larger batch size which leads to a smaller training time.
- DeepLabV3 and UNet are similar in terms of memory usages and training times, but the latter method provides better results.
- By reducing the number of classes, our models consumed less memory, which is allocated into increasing the value of the batch size.
- Attention UNet provides the best results, although it suffers from overfitting.

IV. Project-based Discussion

From the experiments we have conducted, we have gathered the following points:

- Being an image-based project, the memory plays a big role in developing capable models. Images with higher resolution require a lot of memory when uploaded. This slows down the process of training.
- The large number of classes makes the image segmentation process difficult. A possible solution for this is reducing the number of classes, selecting only the ones that appear more often.
- A selection of images from the total of 20.000 images can be made to help with the noise reduction and make more accurate predictions. This happens because there are images from different environments that rarely appear in other photos.
- For a better score, images should be processed at a higher resolution. From our experiments, smaller images lead to overfitted models.
- Larger models would most likely provide better results, but this approach requires better resources (e.g. memory), which were not available to us.
- Data augmentation can be applied to images in order to help the model understand the data and not memorize it. Also, this aids the model to not overfit.

V. Conclusion

In conclusion, this study focused on training machine learning models for semantic segmentation tasks using the ADE20K dataset. Through systematic exploration and evaluation of various models, we gained insights into the challenges and advancements in this domain. By using the ADE20K dataset as a benchmark, we assessed model performance and identified effective architectures for capturing semantic information in diverse scenes.

This experiment can definitely be taken forward and continued to be studied. For further experiments, one can work with larger models, data augmentations, or even Model Soup.

References

[1] Kumar T. Rajamani, Priya Rani, Hanna Siebert, Rajkumar ElagiriRamalingam, Mattias P. Heinrich, Attention-augmented U-Net (AA-U-Net) for semantic segmentation, 2022

[2] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, Daniel Rueckert, Attention U-Net: Learning Where to Look for the Pancreas, 2018

[3] Semantic Segmentation on ADE20K, https://paperswithcode.com/sota/semantic-segmentation-on-ade20k

[4] milesial/Pytorch-UNet, https://github.com/milesial/Pytorch-UNet

[5] Monai - Attention UNet, https://docs.monai.io/en/latest/_modules/monai/networks/nets/attentionunet.html