Transformer-Based Neural Networks for Segmentation and Colorization

Saagar Sanghavi, Jiarui Shan

Final Project: Computer Vision (CS 280)

1 Introduction

Transformer-based neural networks have gained popularity for a variety of Computer Vision tasks, following the success of the first Vision Transformer (ViT) model [1] in 2020. While the original ViT was used primarily for image classification, several later investigations saw the success of using vision transformers for colorization and segmentation tasks. In this project, we investigated the effectiveness of the following Neural Network Architectures:

- Baseline: Standard U-Net Architecture [2]
- Dense Prediction Transformer [3]
- Axial Attention [4] based Colorization Transformer [5]
- Uformer [6]

These networks were trained and evaluated on colorization and segmentation tasks.

The Uformer [6] in particular is a model of interest as it combines the attention mechanism on a U-shaped architecture. However, the task that Uformer was originally trained on was for image restoration, while the original U-Net Architecture was proposed for semantic segmentation on biomedical images. We were most curious to see if we could train the Uformer to perform semantic segmentation or colorization and evaluate its results at inference time.

2 Background: Neural Networks and Transformer-based Models

Deep Neural Networks have seen a huge growth in interest in the past few years as researchers have shown these models to be incredibly powerful for a variety of tasks. Convolutional Neural Networks dominated in the space of Computer Vision since LeNet [7], and though neural networks remained relatively less researched throughout the early 2000s, the success of AlexNet [8] in 2012 reignited research interest in deep neural networks for Computer Vision tasks. In particular, AlexNet proved to be hugely successful on the ImageNet Large Scale Visual Recognition Competition (ILSVRC) by taking advantage of a deep architecture and training on GPUs. Since then, computer vision with machine learning techniques has trended towards deeper models with more parameters.

The classical bias-variance tradeoff in statistics states that the expected squared error of a function class (model) can be decomposed into a variance, squared bias, and irreducible error term. As the model complexity (measured by number of parameters) increases, the bias of the model decreases, but the variance of a model increases. Accordingly, there should be an "optimal middle ground" for the number of parameters in the model which results in the best performance. However, recent results have noted a phenomenon where deeper models with more parameters than data points perform better despite what the classical bias-variance tradeoff might suggest. This phenomenon is termed "double descent": as the number of parameters increase, the test error first declines, then increases again in the "underparameterized regime" before declining further in the "overparameterized regime". The most common explanation for this phenomenon is that extra features have a regularizing effect and prevent overfitting to the data.

The recent successes of deep neural networks for a variety of tasks can be attributed to a number of factors. For one, the availability of large, labeled public datasets has proven to be useful to train models and experiment with different architectures. Furthermore, advancements in computer hardware, parallelism, and low-level optimizations allow us to now take advantage of much more powerful computing capabilities than ever before. This makes training deep models on large datasets feasible today while practitioners in the past

may have been limited by compute resources. Also, the development of automatic differentiation packages like PyTorch and Tensorflow allow researchers to quickly implement and experiment with new architectures, enabling research to be conducted faster. These packages implicitly create computation graphs and automatically perform numerical differentiation and backpropagation of gradients, without the programmer having to hand-calculate the gradients through different layers of the network.

3 Tasks and Datasets

In this project, we trained and evaluated our neural networks on Colorization and Semantic Segmentation tasks. Both Colorization and Semantic Segmentation are pixel-level tasks, and thus require architectural choices that allow for a prediction associated with every pixel.

3.1 Colorization

Colorization refers to the task of filling in the color values for pixels on a greyscale image, and requires that models learn to extract a semantic understanding of the scene and combine that understanding with knowledge of the world. Because colorization of pixels is a contextual task, the training process is usually very data-hungry and the model can often perform unpredictably on new, unseen data. As such, it often requires training for long periods of time on diverse data examples to get the desired results.

The dataset we used to train on the colorization task was the Pascal Visual Object Classes (VOC) dataset [9], which consists of 11,530 images of 20 different classes of objects. Creating training data for colorization simply requires averaging RGB values into a greyscale value (no labels required), so we used the VOC dataset simply for the convenient APIs it offers for loading and storing images.

3.2 Semantic Segmentation

Semantic Segmentation is the task of grouping and labeling every pixel in an image by what object/class it belongs to in the scene.

Classical approaches to semantic segmentation tended to divide the problem into two separate, but related stages: the first stage involves identifying **spatial support**, or which pixels to include as each part of the scene ("segmentation"), and the second stage involves computing **similarity metrics** of the region's statistics with known classes to label (ie. provide "semantics") to each part of the scene. The spatial support stage often took advantage of clustering approaches, such as mean-shift clustering [10] or Spectral Graph techniques [11]. Clusters of similar features often correspond to a single object, so grouping these pixels together was the natural approach to segmenting objects. The second stage typically took advantage of texture-based approaches, such as using pixel histograms, to label the clusters of pixels. One interesting result that many researchers found was that making progress on identifying spatial support often resulted in the task of labeling segmented portions of the image becoming a lot easier. Modern approaches to semantic segmentation have, as usual, found great success using Neural Networks for end-to-end training.

The dataset we used to train on the semantic segmentation task was the Hypersim dataset [12], which is a synthetic dataset of indoor scenes. Because semantic segmentation requires data with per-pixel labels, synthetically generated data is often easier to obtain and more accurate then trying to have human labelers manually segment parts of an image and label them.

A reason we choose semantic segmentation in addition to colorization is because they are two very different pixel-level prediction tasks. Colorization is a regression problem whereas segmentation is a classification problem. Furthermore, the output domain of colorization is similar to the input domain (both are RGB images, except the input RGB values are identical), but for segmentation, the output is very different from the input domain.

4 A Survey of Current Methods and Approaches

Convolutional Neural Networks (CNNs) typically are the standard choice when it comes to deep learning for Computer Vision. Convolutional layers take advantage of weight-sharing and shift-invariance to make the best use of fewer parameters, which has a favorable inductive bias that exploits the spatial locality of features in an image. In addition, pooling operations allow the later layers to have an increasing receptive field size compared to earlier layers, and fully connected layers after flattening the feature maps at the end

of the network allows the CNN to incorporate information from all parts of the image when making a final prediction. Though originally proposed for object classification of a single image, convolutional layers are used in a variety of architectures to achieve great results on object localization, detection, and semantic segmentation tasks as well.

The common challenge with pixel-level prediction is that because an image may be very large/high resolution, if the neural network architecture never performs downsampling/pooling, a single forward pass to generate a prediction for every pixel would likely be too computationally expensive. This was the problem that the U-Net [2] aimed to solve by having an architecture that performs downsampling by max pooling ("encoding") before doing visual processing on low-resolution feature maps, then performs upsampling by transpose convolutions ("decoding") to eventually recover a segmentation map. The feature map at each level of the decoder also concatenates the output from the encoder side at the same level, as shown in Figure 1.

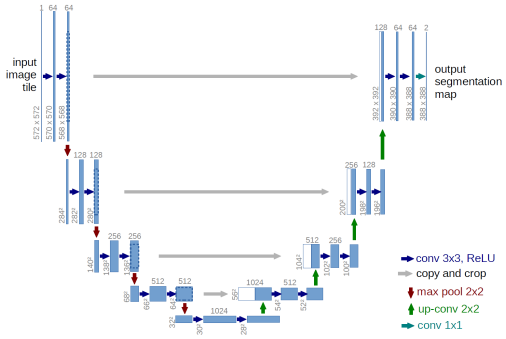

Figure 1: U-Net Architecture. Source: [2]

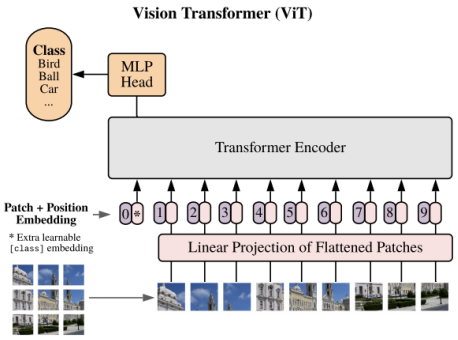

Figure 2: Vision Transformer. Source: [1]

The Transformer [13] is a neural network architecture that was originally proposed for language sequence modeling. This architecture uses multi-headed self-attention to allow the model to learn long-range dependencies between sequence elements. While the motivation for Attention and Transformers came from the bottleneck problem in RNNs for sequence modeling, later investigations found the Transformer architecture to be effective for vision tasks as well, with the Vision Transformer [1] being a seminal work that achieved state-of-the-art results on image classification. The Vision Transformer treats an image as a sequence of 16x16 patches, and uses a projection of the flattened patches as the "tokens" that are fed into a Transformer Encoder block. An additional dummy [class] token with a learnable embedding is also fed into the encoder, and its corresponding output token is passed through a feedforward MLP head to determine the final classification of the image, as shown in Figure 2. An potentially useful characteristic of the vision transformer is that unlike convolutional layers, the receptive field of a single unit after a transformer layer includes the entire image. As a result, transformer-based architectures have an inductive bias towards globally coherent predictions.

Since the success of the Vision Transformer in 2020, several other projects have made attempts at applying attention-based architectures to vision tasks. Naturally, treating each pixel in an image as a token and performing attention with all other pixels in the scene can be costly and computationally infeasible. As a result, attention-based architectures for vision take advantage of different techniques to reduce the amount of computation required.

4.1 Dense Prediction Transformer

The simplest architecture choice we investigated was the Dense Prediction Transformer (DPT) [3], which is based on the encoder-decoder design and uses a Transformer encoder block as the basic computational block for the visual encoder. However, unlike the U-Net architecture, the backbone does not perform downsampling between sucessive transformer layers and the tensors remain the same size throughout all phases of the image processing. On the decoder side, DPT uses a "Reassemble" block, which concatenates the embeddings of the patch tokens and applies a linear transformation to the reconstructed representation before progressively fusing the representations with the reassembled outputs from previous layers. This architecture is illustrated in Figure 3.

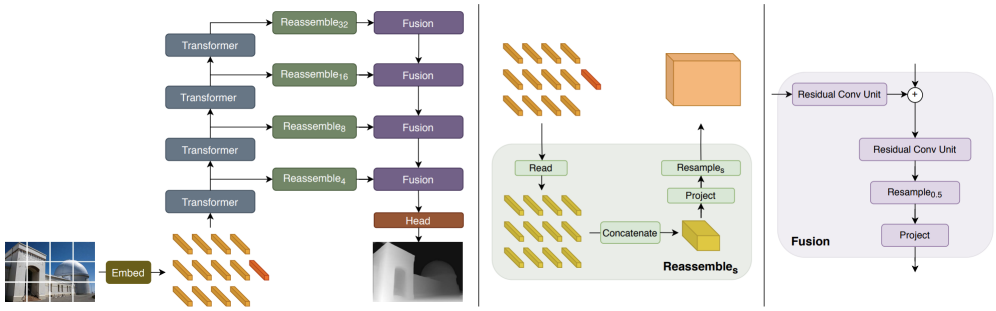

Figure 3: Dense Prediction Transformer Architecture. Source: [3]

4.2 Axial Attention (Colorization Transformer)

Another attention-based architecture that we experimented with was the Colorization Transformer [5], which uses Axial Attention [4]. Axial Attention allows the model to treat each pixel as an individual token, but each pixel only attends to other pixels in the same row and column as a way to reduce the total amount of computation required, as shown in Figure 4. As a result, the Axial Attention block still captures a receptive field the size of the full image after two layers. Furthermore, Axial Attention can be parallelized and made more efficient using modern TPUs and accelerators, and thus can colorize images much faster than previous approaches.

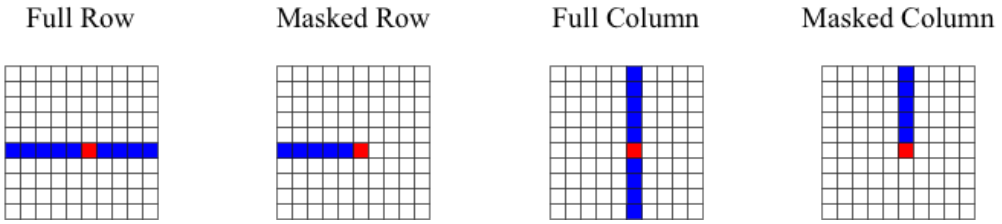

Figure 4: Axial Attention for a single pixel. Source [4]

The colorization transformer has a slightly more advanced architecture that consists of 3 modules: an autoregressive colorizer, a color upsampler, and a spatial upsampler, each of which are optimized independently of one another. A schematic of the architecture is shown in Figure 5.

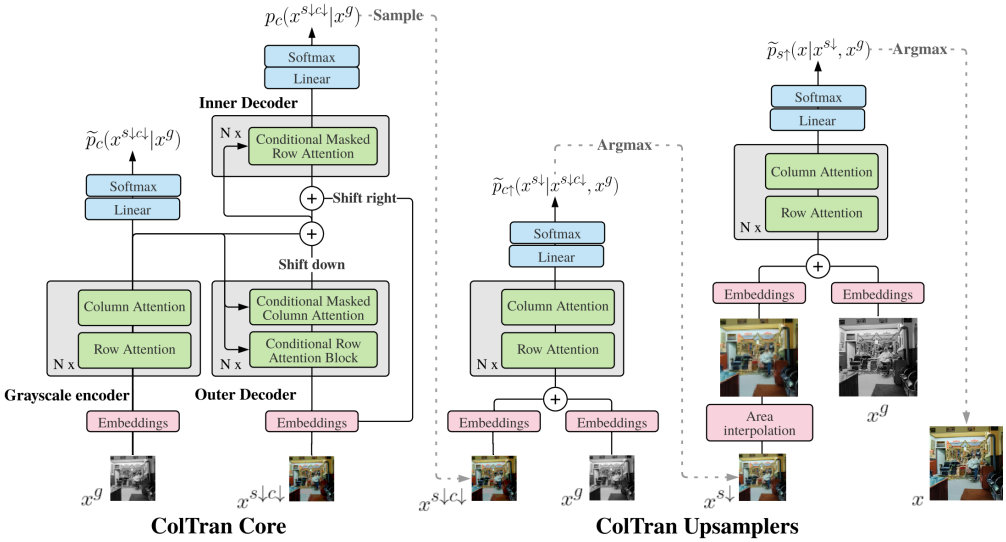

Figure 5: Colorization Transformer Architecture. Source: [5]

In the autoregressive colorizer module (ColTran Core), the model performs conditional masked row- and column-attention on a downsampled version of the greyscale image to autoregressively generate samples of pixel-level predictions. Then, the color upsampler combines the embedded output of the autoregressive colorizer with an embedding of the greyscale image and performs row- and column attention to get an emedding of the image in a richer color space. Finally, the spatial upsampler performs bilinear interpolation on the colorized image, combines an embedding of the greyscale image again, and passes it through one last layer of axial attention to get the final output.

4.3 Uformer

The final model architecture that we investigated was the Uformer, which is a U-shaped architecture that uses a Locally enhanced Window (LeWin) Transformer block to perform window-based self-attention (as opposed to global self attention). This architecture was originally proposed for restoration from corrupted images, and for removing undesired image degradation like noise, rain, or blurring. The Uformer closely models the structure of the U-Net Architecture, keeping the hierarchical visual encoder-decoder structure and residual concatenations; however, all convolution operations are instead replaced with the LeWin Transformer block. The architecture is illustrated in Figure 6.

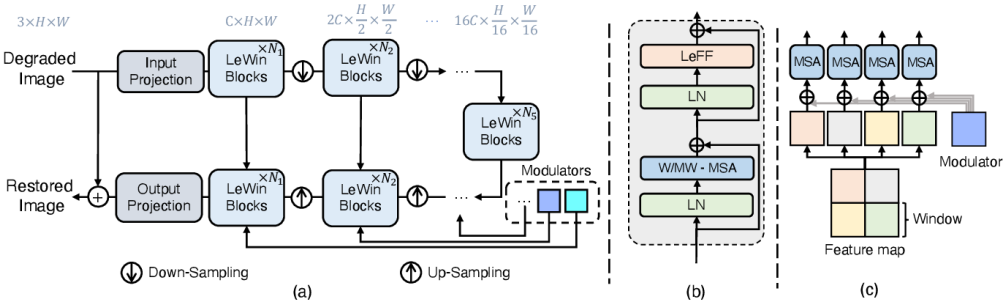

Figure 6: (a) Uformer, (b) LeWin Block, (c) Windowed Multi-head Self Attention (W-MSA). Source: [6]

The Locally enchanced Window (LeWin) block, as shown above, is similar to a normal Transformer Encoder block in that it has a multi-headed self-attention module followed by a Feedforward module, both of which have residual connections around them. However, it differs slightly in the position of the LayerNorm, as well as how the Self-Attention and Feedforward Modules are computed. The model uses Windowed Multi-head Self Attention (ie. W-MSA), meaning that instead of attending to all other tokens like the original transformer, the self-attention is performed within non-overlapping windows, and each pixel only attends to other pixels within the same window. In addition, the usual Fully-Connected layer in the Transformer Encoder block is replaced with a Locally enhanced Feed Forward block (LeFF), which arranges the tokens into an image, performs a 3x3 depthwise convolution, then converts the image back into the sequence of tokens.

5 Experimental Setup

For each of the two tasks (Colorization and Semantic Segmentation), we set up a separate neural network for each of the 4 architectures we were interested in comparing: the U-Net baseline, Axial Transformer, Dense Prediction Transformer, and Uformer. While some of the architectures worked as originally described on a task (for example, the U-Net on the semantic segmentation task), most of the architectures required some modifications to suit the task at hand (for example, a classification head on top of each pixel prediction in the case of adapting the Axial Transformer for segmentation).

For the colorization task, we trained the model used the Charbonnier Loss Function [14], given by $\ell(x, \hat{x}) = \sqrt{(x - \hat{x})^2 + \epsilon}$, where $\epsilon = 10^{-6}$ or another small constant. This loss function is sometimes called a "pseudo-L1" loss or a "pseudo-Huber" loss as it is very similar to just taking the abolute value, but smoothed for small values of $|x - \hat{x}|$ and thus differentiable everywhere.

For the semantic segmentation task, we trained the model using the Negative Log Likelihood (NLL) Loss.

We trained all the models for 300 epochs, which was long enough for the loss values to converge for all the models.

6 Results and Analysis

6.1 Results on Colorization

After training all models for 300 epochs, we use the L1 loss to measure how close the models' predictions are relative to the ground truth for both the training set and validation set. A model's L1 loss on the training set indicates its fitting power, and the L1 loss for the validation set is a measurement of how well the model actually learned meaningful and generalizable knowledge about the data from training.

Model	# of Parameters	Training Loss	Validation Loss
U-Net	31037763	0.0339	0.0487
DPT	112501422	0.0542	0.0696
Axial Transformer	19944323	0.0404	0.0479
Uformer	20628317	0.0436	0.0459

Table 1: Training and Validation L1 Losses on Colorization task

Based on those statistics, we are able to draw several meaningful conclusions. First, the U-Net has the best fitting power compared to all three transformer-based models, indicated by having the lowest training loss after the same amount of training. This could be the case because the U-Net model has roughly 1.5 times the number of parameters as the Axial Transformer and the Uformer, with a total of about 3×10^8. (The DPT does have more parameters than the U-Net, but we will discuss why it might have failed the colorization task later). Another possible explanation is that the U-Net is relatively shallow compared to the other models (because one CNN layer is shallower than one attention block), so its weights are updated more efficiently than transformer-based models. In addition, the U-Net has the fastest inference time among all the models, at around 0.023s per 224x224 image.

However, U-Net would not be a good option for colorization unless more regularization techniques are put into place. Although U-Net does have the fitting power to recover the details of the structure of the input, it does not have a solid understanding of how to colorize in image. As show in the example below, U-Net has the worst interpretation of what color the sky should be among all 4 models. This could be due to the convolution operation allowing each pixel to only attend to its neighbors, so U-Net's awareness of a scene as a whole could be relatively limited.

Figure 7: Ground Truth Color Image

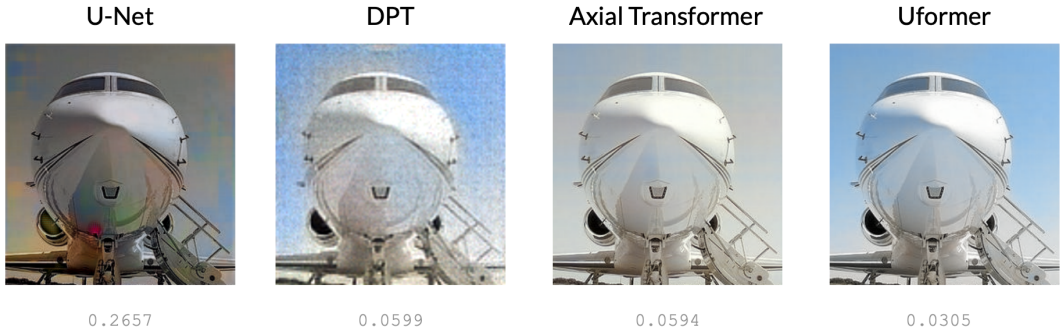

Figure 8: Colorization performance of our 4 candidate models on a validation image, which illustrates the points we made in the analysis.

Another (and arguably the most important) finding from our experiments on the colorization task is that the Uformer has the best overall best performance on the evaluation dataset. Given that all models are trained from randomly initialized weights without any pretrained components and we did not use any additional data augmentation or regularization techniques, Uformer appears to be the most robust in terms of generalizing to unseen data, as illustrated by the example below. We believe that the architectural design of Uformer is naturally suitable for tasks such as colorization, where the output can be approximated by computing deltas from the input. Uformer has residual connections built-in to every scale, and is essentially always computing how much to nudge the inputs to predict the output rather than predicting the absolute RGB values. In particular, in the final layer of Uformer, the input is added to the network's output. This is an advantage in colorization because even before training happens, Uformer could already recover the input image in full resolution (since weights are initialized with zero mean) and just needs to figure out how each pixel should be colorized. In contrast, other networks need to spend additional effort to adjust its weights to recover the structure of the input image, in addition to colorizing it.

Finally, we want to bring up our observation for the Dense Prediction Transformer. Specifically, we find that the DPT is inherently more noisy than the other models, and is unable to perfectly recreate the input scene without considering the colors. Our main explanation for this is that since DPT uses the Vision Transformer backbone, it treats an image as a sequence of patches for the purpose of self-attention. However, each image patch is first flattened through a linear projection into a fixed-length vector in order to go through the attention layer and is later flattened back to an image patch. We believe that this process is potentially lossy. Although the fixed-length vector's total dimension is equal to the total dimension of the image patch ($768 = 3 \times 16 \times 16$), meaning that there do exist vector encodings that keep the full information in the image patch, the 2D structure of the image is destroyed during projection and unprojection, and a really good mapping needs to be learned in order to preserve all the information in the input. With limited training, such mapping would be infeasible to create, so DPT produces noise in the output on both training and validation images.

6.2 Results on Segmentation

We trained all models for 2 weeks on the HyperSim segmentation dataset, and evaluated the models based on the Negative Log-Likelihood (NLL) loss on the validation set and the Top-1 accuracy on the validation set. Again, the Training loss at convergence represents how well the model was able to fit the training data, while the validation loss and eval accuracy represents how well the model generalized to unseen data.

Model	# of Parameters	Training Loss	Validation Loss	Eval Top1 Accuracy
U-Net	31040168	0.1423	3.3514	45.3%
DPT	112520440	0.335	3.576	36.4%
Axial Transformer	19990056	0.5221	2.1119	41.0%
Uformer	20649666	0.3382	2.0002	46.6%

Table 2: Training and Validation NLL Losses on Segmentation task

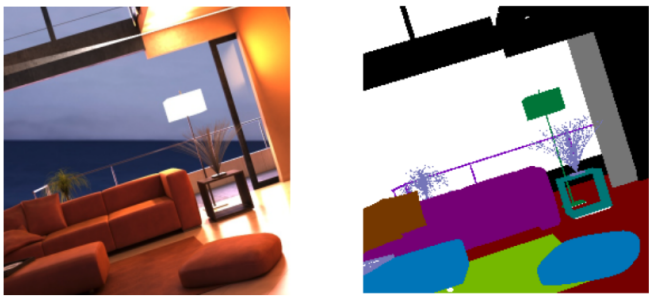

Figure 9: Ground Truth Segmentation Input/Output pair

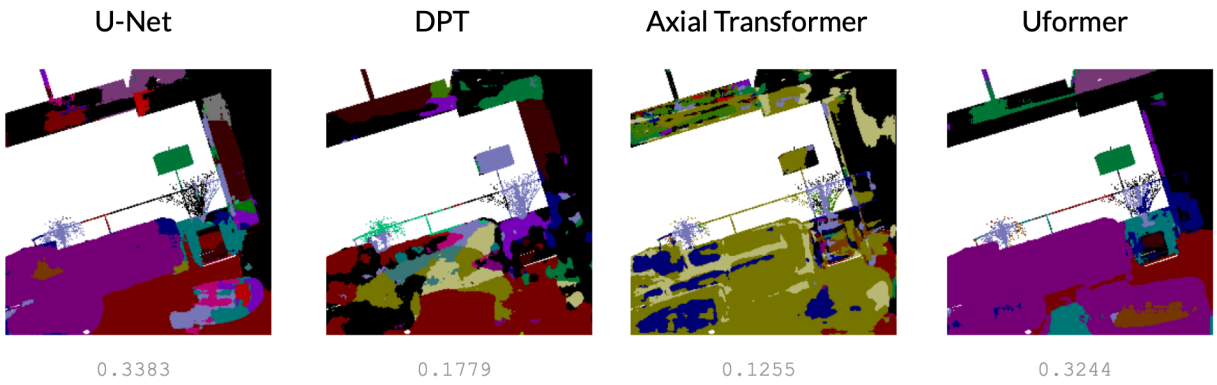

Figure 10: Segmentation performance of our 4 candidate models on the validation image.

We can make a few observations based on the loss statistics and the chosen example validation image. First, Dense Prediction Transformer remains to be the worst model in terms of the validation loss or the top-1 accuracy. The reason why this could be the case is probably similar to our reasoning for its performance on colorization, that the DPT has to go through a linear projection and inverse projection process, and it's easy to lose information during those transformations.

Interestingly, the Uformer continues to be the best model in our segmentation task in terms of both the top-1 accuracy and the validation loss. This suggests that the Uformer architecture is general enough (as a transformer-based replacement of U-Net) to be used for both pixel-level classification and regression tasks. In the example image (Figure 8), we see that Uformer's segmentation predictions are relatively sharp compared to the other methods.

Our observation for U-Net on our segmentation task confirms our previous conclusion about U-Net, that it has a strong fitting power but could easily overfit. In Table 2, we continue to see U-Net being the model with the lowest training loss, but the validation loss is still higher. Ideally, data augmentation techniques should be used on U-Net for segmentation tasks in practice.

Finally, the Axial Transformer seems to have a slightly harder time to fit the data compared to other models. This could be because segmentation task requires awareness of the scene in 2D, but Axial Transformer only performs self-attention in rows / columns of pixels, so each pixel only has limited access to its surroundings in 1D. In particular, a pixel cannot attend to pixels that are diagonal to it within a single layer. Since Axial Transformer attains the lowest training loss among all 3 transformer-based architectures for colorization, we believe that Axial Transformer is just better suited for colorization, and that colorization and segmentation requires different understandings of the input image.

7 Conclusion

In this project, we adapted several transformer-based models for semantic segmentation and colorization and compared them to the Convolutional U-Net baseline. We found that the Uformer performed well on both tasks, and generalized better than other methods. Though the architecture is relatively new, we expect it to gain a lot more traction in the computer vision literature in the upcoming years.

Our code and models can be found at the following repository: `https://github.com/TheMoon2000/cs280-final-project`

References

[1] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929 (2020). arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[3] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. "Vision Transformers for Dense Prediction". In: *CoRR* abs/2103.13413 (2021). arXiv: 2103.13413. URL: https://arxiv.org/abs/2103.13413.

[4] Jonathan Ho et al. "Axial Attention in Multidimensional Transformers". In: *CoRR* abs/1912.12180 (2019). arXiv: 1912.12180. URL: http://arxiv.org/abs/1912.12180.

[5] Manoj Kumar, Dirk Weissenborn, and Nal Kalchbrenner. "Colorization Transformer". In: *CoRR* abs/2102.04432 (2021). arXiv: 2102.04432. URL: https://arxiv.org/abs/2102.04432.

[6] Zhendong Wang et al. "Uformer: A General U-Shaped Transformer for Image Restoration". In: *CoRR* abs/2106.03106 (2021). arXiv: 2106.03106. URL: https://arxiv.org/abs/2106.03106.

[7] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[9] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 2012.

[10] D. Comaniciu and P. Meer. "Mean shift: a robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5 (2002), pp. 603–619. DOI: 10.1109/34.1000236.

[11] Ulrike von Luxburg. "Spectral Clustering". In: *Statistics and Computing* 17.4 (2007).

[12] Mike Roberts and Nathan Paczan. "Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding". In: *CoRR* abs/2011.02523 (2020). arXiv: 2011.02523. URL: https://arxiv.org/abs/2011.02523.

[13] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[14] Jonathan T. Barron. "A More General Robust Loss Function". In: *CoRR* abs/1701.03077 (2017). arXiv: 1701.03077. URL: http://arxiv.org/abs/1701.03077.