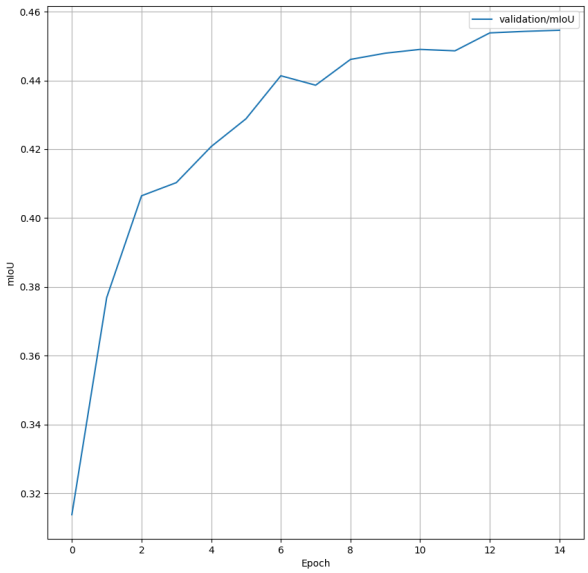
# 1 Segmentation

## 1.1 Linear Head




(a) mIoU vs Epochs


(b) Epoch 0


(c) Epoch 4


(d) Epoch 9


(e) Epoch 14

Figure 1: Performance of Linear decoder head

## 1.2 Convolutional Head

### 1.2.1 Architecture

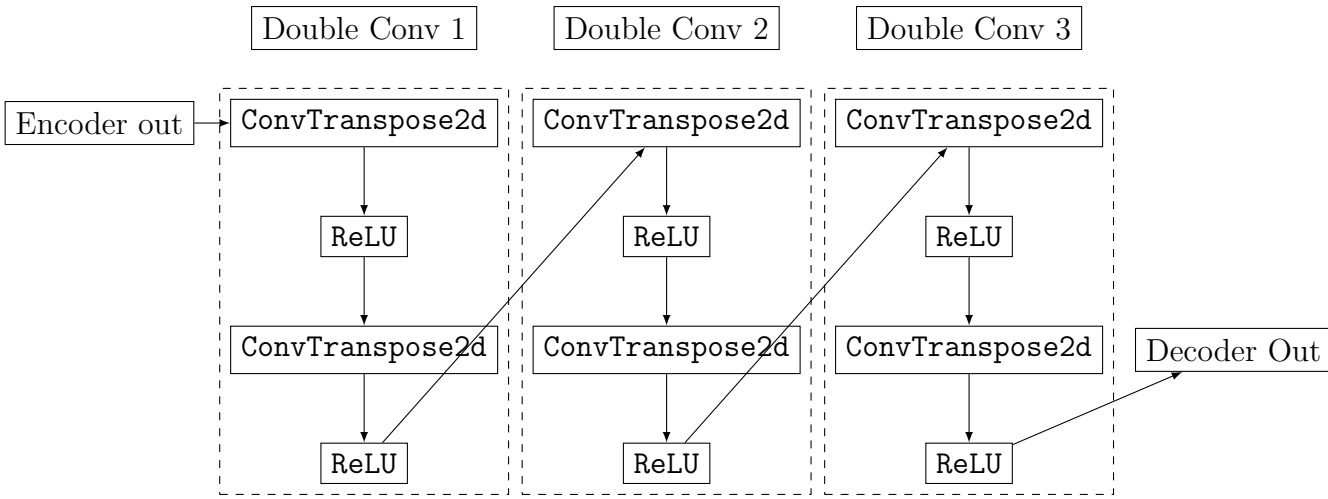
The architecture of the convolutional head contains 3 sequential wrappers each having two **ConvTranspose2d** layers, with both followed by **ReLU** activations before their output is passed to the next Double convolution layer. The Transposed convolution performs the opposite of convolution and increases the spatial resolution of the feature maps. Each double convolution block has the following structure:

- **ConvTranspose2d** layer that doubles the input channels
- **ReLU** activation function.
- **ConvTranspose2d** layer that halves the doubles input channels
- **ReLU** activation function.

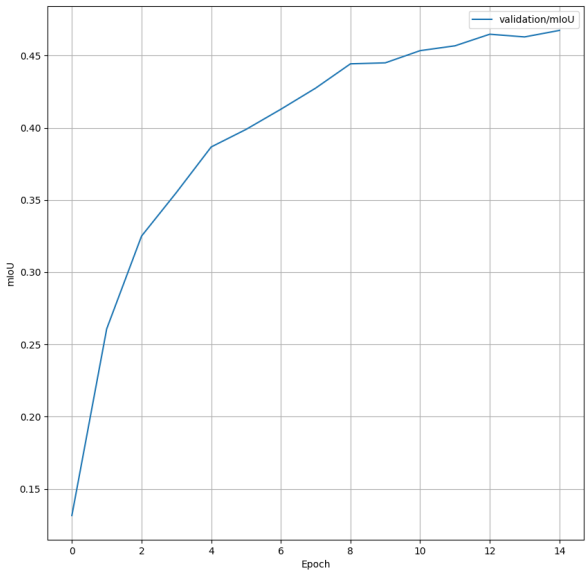
The structure of the decoder is as follows:

- The first double convolution block takes input with Encoder embedding dims and produces output with Encoder embedding dims $\times$ 2.
- The second double convolution block takes input with Encoder embedding dims $\times$ 2 and produces output with Encoder embedding dims.
- The third double convolution block takes input with Encoder embedding dims and produces output with the number of classes in the training dataset.

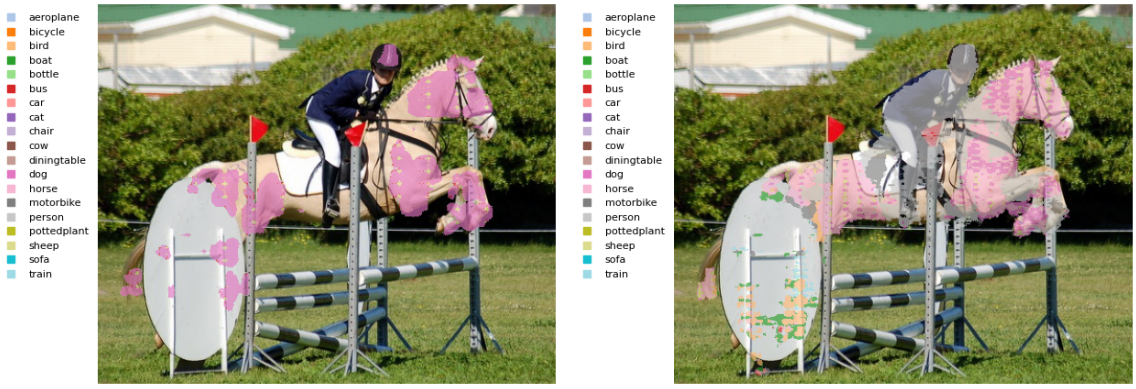
**Convolutional head architecture for decoder**



### 1.2.2 Validation plot and qualitative results



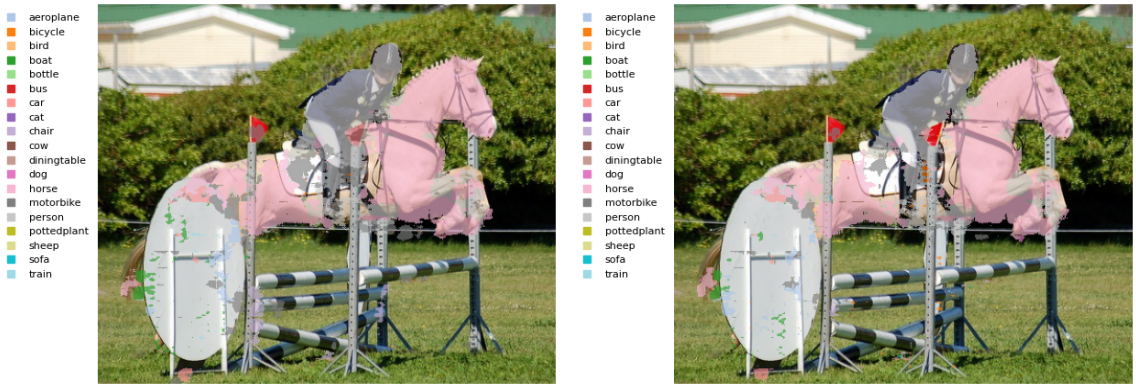

(a) mIoU vs Epochs




(b) Epoch 0




(c) Epoch 4




(d) Epoch 9



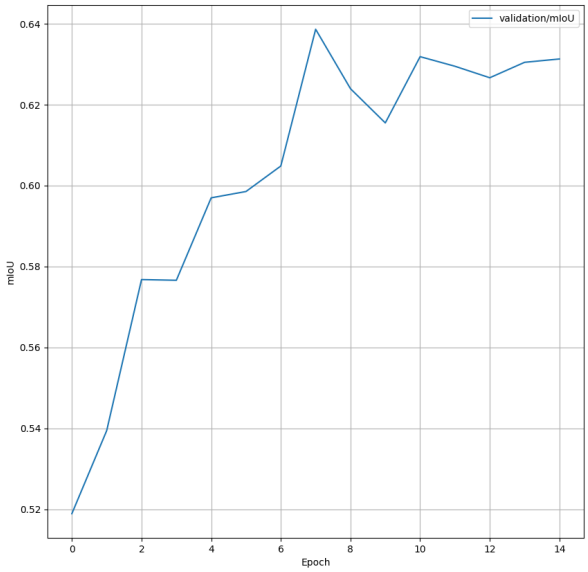

(e) Epoch 14

Figure 2: Performance of Convolutional decoder head

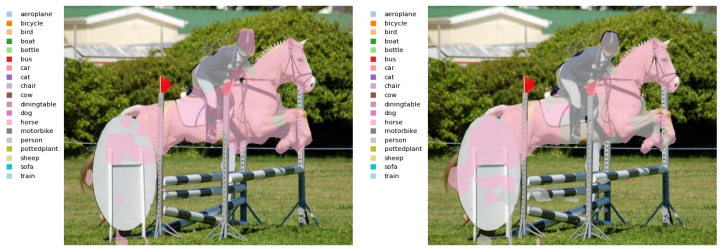
The model seems to perform better than the Linear decoder head, as there are more spatial features due to the upsampling that happens across the three blocks. Qualitatively, there are lesser spots in the images that correspond to different classes, and we see in places of homogeneity, that there is no misclassification. This could be due to the increased spatial sampling, that takes information about the context from farther away than the linear head.

## 1.3 Transformer Head

The transformer head performs leagues better than either of the two previous heads. The attention mechanism helps in focusing on the more important parts with respect to the context, leading to better results earlier on in the training.



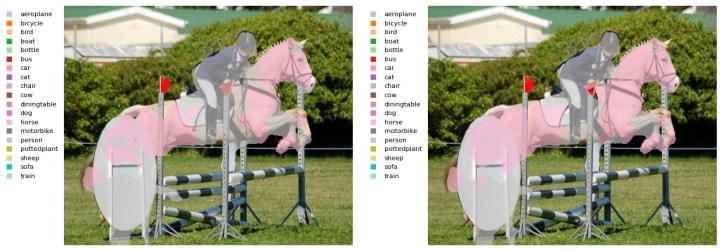

(a) mIoU vs Epochs




(b) Epoch 0




(c) Epoch 4




(d) Epoch 9

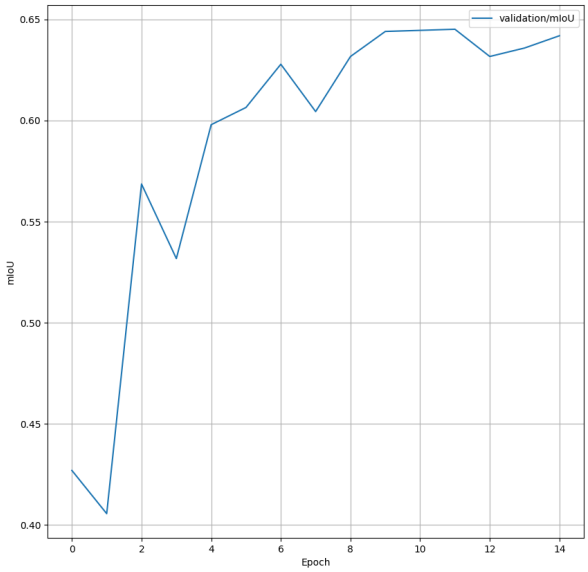



(e) Epoch 14

Figure 3: Performance of Transformer decoder head without Query-Key sharing

### 1.3.1 Shared Queries and Keys

This is the content of the shared queries and keys sub-section.




(a) mIoU vs Epochs


(b) Epoch 0




(c) Epoch 4


(d) Epoch 9




(e) Epoch 14

Figure 4: Performance of Transformer decoder head with Query-Key sharing

My understanding for why this works better is, that the usage of the same queries for the keys, will let the model search for its intrinsic learned relation for the data throughout the other patches. This lets it share the information that it learned from one patch and use it for another, thus increasing the spatial context of the image.

## 1.4 Putting It All Together

The following table puts together my findings. The shared Query-Key attention mechanism led to an interesting observation. In the image given for the validation results, even though the legs of the horse are occluded, the model seems to make a somehwat accurate prediction of where the horses legs would be and classes them as belonging to the horse. This could mean that just like a human does, the model also has learned to identify parts of the object that might be occluded. Once again this might be an extremely specific case owing to the image i have used, but it still interests me. I will have to test further to understand whether its really what happening.

Table 1: Decoder Architectures and Average IoU Scores

| **Decoder Architecture** | **Average IoU Score** |
|---|---|
| Linear | 45.46 |
| Convolutional | 46.47 |
| Transformer | 63.13 |
| Transformer with Key Sharing | 64.20 |

# 2 Image-Text

## 2.1 Image Captioning

### 2.1.1 Temperature

The effect of temperature is that a lower value reduces the amount of randomness that the model has while selecting the words. While a higher temp may be good in generating more predictive captions, it also leads to words that do not add any sense to the output to also be generated while captioning. So naturally it leads to the effect that lowered temperature makes the model more conservative, hence increasing likeness to the reference caption. I think this explains a higher BLEU score for lower temperature.

### 2.1.2 Experimentation

The following table summarises different hyperparameter settings along with the ones already included in the assignment sheet. I didnt know what prompts to use apart from the default one, as that seemed to fit best with the images that were given, hence I opted to keep the prompts the same for all the runs. I managed to get a BLEU score higher than the greedy sampling baseline.

A table containing the reference captions and the predicted captions for the best performing hyperparam model is given below. The effect of the extremely low temeprature is evident, as there is a decrease in the desciptiveness when compared to the refrence captions:

Table 2: Method, Hyperparams, and BLEU Score

| **Sampling method** | **Batch Size** | **Temperature** | **TopK Value** | **BLEU Score** |
|---|---|---|---|---|
| Greedy | 16 | - | - | 0.277 |
| Top-k | 16 | 1.0 | 50 | 0.053 |
| Top-k | 16 | 0.7 | 50 | 0.1056 |
| Top-k | 64 | 0.1 | 50 | 0.2094 |
| Top-k | 64 | 0.1 | 20 | 0.2499 |
| Top-k | 64 | 0.05 | 50 | 0.2756 |
| Top-k | 64 | 0.05 | 20 | 0.2764 |
| Top-k | 64 | 0.01 | 50 | **0.2769** |

Table 3: Comparison of Reference and Predicted Captions

| **Reference Captions** | **Predicted Captions** |
|---|---|
| a plane on the tarmac | a plane at an airport with a gate |
| two white and red trains parked next to each other | two trains parked on the tracks |
| a boat docked in a waterway next to a house | a boat on the water near a dock |
| a black and white photo of a train at a station | a train at a train station |
| a group of cyclists racing on a track | a group of people riding bikes |
| two sheep laying in the grass | two sheep in the grass together |
| a computer is sitting on a desk next to a monitor | a computer and a monitor on a desk |
| a man and a woman posing for a picture | me and my friend, I'm in the train |
| a horse standing next to a green slide | a horse that is standing in the grass |
| a woman holding up a bottle of vodka | me with a bottle of vodka |
| a couch in a living room | a living room with a couch and a chair |
| two cows laying on the beach | two cows on the beach with the ocean in the |
| a cow with a blanket on its back | a cow laying down in a pen |

## 2.2 Image Text Retrieval

### 2.2.1 Eval Captioning

The eval captioning method was filled out and worked just fine. The following are the scores

Table 4: Text Retrieval Evaluation Results

| **Metric** | **Value** |
|---|---|
| i_r1 | 0.54 |
| i_r5 | 0.80 |
| i_r10 | 0.91 |
| i_medr | 1.00 |
| i_meanr | 4.07 |
| t_r1 | 0.53 |
| t_r5 | 0.81 |
| t_r10 | 0.89 |
| t_medr | 1.00 |
| t_meanr | 5.21 |

### 2.2.2 Train Retrieval

The following are the results for training the retrieval projection layers, with both the experimentation and default modes as given in the assignment. I couldnt improve over the baseline, and will have to test further to see what settings work. From the limited amount of compute that I had while running the experiments, I gather that a lower weight decay with a lower temperature would lead to better performance, owing to the conservativeness while selecting words via the temperature, and more information being saved due to the lower weight decay.

Table 5: Model Evaluation Results

| **Initialization** | **Hyperparams** | | | | | **R1 Score** |
|---|---|---|---|---|---|---|
| | **LR** | **Weight Decay** | **Temp** | **Epochs** | **Batch Size** | |
| Finetune | $1 \times 10^{-5}$ | 0 | 0.1 | 3 | 16 | **0.605** |
| Random (default) | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ | 0.1 | 5 | 16 | 0.439 |
| Random (exp 1) | $2 \times 10^{-4}$ | $1 \times 10^{-2}$ | 0.2 | 5 | 32 | 0.361 |
| Random (exp 2) | $2 \times 10^{-3}$ | $1 \times 10^{-2}$ | 0.2 | 5 | 64 | 0.413 |
| Random (exp 3) | $2 \times 10^{-4}$ | 0.01 | 0.1 | 5 | 64 | 0.30 |
| Random (exp 4) | $2 \times 10^{-4}$ | $1 \times 10^{-5}$ | 0.1 | 10 | 64 | 0.404 |