

Final Project: Stable Diffusion and Image Captioning  
Calvin Pugmire  
CS 474 Section 001

## Stable Diffusion

### The Problem

The task for the first half of my project was to create a fine-tuned Stable Diffusion LoRA model for generating customized clone troopers. This is a generative problem: How do you generate an image based on a given text? It uses supervised learning through training on labeled images. When it comes to useful background knowledge, I have general DNN knowledge from class, experience using Stable Diffusion, and Star Wars expertise. There has been one other clone trooper model released, but it performs rather poorly overall. Furthermore, it only contains a limited number of legions and is modeled specifically after "Star Wars: The Clone Wars" troopers.

### The Dataset

The dataset I used for this model came from clone trooper images gathered from across the web, paired with captions that I made myself. Various Stable Diffusion artists have been looking for an easy way to make customized clone troopers since August of 2022, and nobody seems to have made a satisfactory one as of yet. The data I used consists of around 200 images of clone troopers, each paired with a text-based set of descriptions.






### The Technical Approach

The fine-tuning process for stable diffusion models has become rather streamlined: You gather images, create labels for them, choose a base model to fine-tune, adjust various training parameters in a training algorithm like Kohya-SS, and then run said algorithm on your chosen subjects. The base model I used was v1-5-pruned. It is a well-rounded, general-purpose Stable Diffusion model. The training algorithm I used was Kohya-SS. I did not have (or need) access to its internal workings, as this part of my project focused on the research, data preparation, and testing+analyzing aspects of deep learning. The data's training/testing split was also handled by the Kohya-SS algorithm. The model includes 87 hyperparameters. It also uses an AdamW8bit optimizer. The topology used by the model is a Latent Diffusion Model (fundamental for all Stable Diffusion models), utilizing a U-Net (for denoising) and an autoregressive model (for diversity and overfitting prevention). It also uses pre-trained weights, which come from the v1-5-pruned base model.

### The Results

The model's performance is not reported by the Kohya-SS algorithm. I did not have (or need) access to it, as this part of my project focused on the research, data preparation, and testing+analyzing aspects of deep learning. However, after properly cleaning and labeling my data, the model's performance was satisfactory. I was also able to ascertain the fact that it did not overfit my data. After thorough testing, I was able to confirm that the model responds well to diverse prompts and also supplies diverse solutions to repeated vague prompts. I had three iterations of my model. The first iteration lacked labels describing the type of shot that was taken (full body, portrait, etc.) and would create an ugly mismatch of an image when asked for something larger than 512x512. The second iteration lacked labels describing the background (hallway, forest, etc.) and would create repetitive backgrounds for nearly all of the images. The third iteration had no notable problems and performs well. I have solved the problem I set out to solve. I have made a Stable Diffusion LoRA model that can successfully make customized clone trooper images based on submitted prompts.

### Some Outputs:

				
Special ops trooper in the nighttime city.	Sniper in the mountains.	Bright orange (for fun).	Yes ma'am.	Christmas trooper.

## Image Captioning

### The Problem

The task for the second half of my project was to create an image captioning transformer-based DNN using the Flickr8k or COCO dataset. This is a generative problem: How do you generate a text based on a given image? It uses supervised learning through training on already-captioned images. When it comes to useful background knowledge, I have general DNN knowledge from class. In the past, both CNNs and RNNs have been made to caption images. These models are simple and effective, but can struggle with long-range dependencies.

### The Dataset

The datasets I used for this model were the Flickr8k and COCO datasets. The Flickr8k dataset was published by adityajn105 on Kaggle.com. The COCO dataset was published by various collaborators on COCOdataset.org. Many deep learning engineers use these datasets to train various DNN networks, especially those for labeling images, captioning images, and generating images. The Flickr8k dataset consists of 8091 images, each paired with 5 human-made captions. The COCO dataset works similarly, with the difference being that it contains 82783 images.






### The Technical Approach

Transformer models are generally the best solution when it comes to image captioning. They employ a combination of self-attention and encoder-decoder attention to capture long-range dependencies within an image and its generated caption, leading to more coherent and detailed descriptions. Naturally, I decided to use this approach. The model I used was an encoder-to-decoder (image-to-text) transformer model. For each epoch, the algorithm loops through each batch of training image+caption pairs. The image is fed into the transformer to generate a caption, and then the generated caption is compared to the correct caption to compute the loss. Gradients are then made and applied to the transformer by the optimizer (via backpropagation). There was no test/training split. Because all of the images had five different captions (and were therefore well-rounded), all of the image+caption combinations could be used as training data. The model utilized 13 hyperparameters. It also used the Adam optimizer. The model's topology was an Encoder-Decoder Model (fundamental for all transformer-based models), utilizing self-attention, encoder-decoder attention, positional encoding, and feed-forward layers. It also utilized pre-trained weights from a Vision Transformer (ViT) model.

### The Results

Image captioning transformer models use cross-entropy loss and accuracy to measure performance. When training on the Flickr8k dataset, the model reached a (local) optimum of loss=5.2 and accuracy=0.19. When training on the COCO dataset (roughly 10x larger), it reached a (local) optimum of loss=3.69 and accuracy=0.36. The model did not overfit my data, as it correctly captions images containing diverse subjects. I had four iterations of my model. The first iteration used a dataset class like the one in Lab #8, which broke the algorithm in multiple, time consuming ways and made it ridiculously difficult to modify and debug. The second iteration was trained on the Flickr8k dataset (8091 images), which was too small and stopped at accuracy=0.10 (with captions that repeated 1-3 words incessantly). The third iteration also used Flickr8k but was fine-tuned to maximize performance, but it still stopped at accuracy=0.19 (with somewhat dismal captions). The fourth iteration was instead trained on the COCO dataset (82783 images) and reached accuracy=0.36 (with satisfactory captions). I was able to make progress on the problem I set out to solve. I have made an image captioning transformer-based model (using the COCO dataset) that can correctly caption submitted images.

### Some Outputs:

				
'a' 'stop' 'light' 'is' 'lit' 'up' 'on' 'a' 'street' 'light'	'a' 'dog' 'is' 'riding' 'on' 'a' 'surfboard' 'in' 'the' 'ocean'	'a' 'baseball' 'player' 'throwing' 'a' 'ball' 'on' 'a' 'field'	'a' 'large' 'elephant' 'is' 'standing' 'in' 'front' 'of' 'a' 'fence'	'a' 'train' 'is' 'parked' 'on' 'the' 'train' 'tracks'

Hour Log:

Day:	Description:
10/23	2 hours: -Stable Diffusion model. -Reading+research: <a href="https://www.youtube.com/watch?v=j-So4VYTL98">https://www.youtube.com/watch?v=j-So4VYTL98</a> > Installed Kohya SS Trainer.
10/24	1 hours: -Stable Diffusion model. -Prep work: Dataset collection.
10/31	3 hours: -Stable Diffusion model. -Prep work: Dataset cleaning+preparation.
11/6	2 hours: -Stable Diffusion model. -Main work: Coding+debugging.
11/9	0.5 hours: -Image captioning model. -Main work: Coding+debugging.
11/11	0.5 hours: -Image captioning model. -Main work: Coding+debugging.
11/13	0.5 hours: -Image captioning model. -Main work: Coding+debugging.
11/16	1 hours: -Image captioning model. -Main work: Coding+debugging.
11/18	1 hours: -Image captioning model. -Main work: Coding+debugging.
11/21	2.5 hours: -Image captioning model. -Main work: Coding+debugging.
11/22	3.5 hours: -Image captioning model. -Main work: Coding+debugging+testing+analyzing.
11/23	1 hours: -Image captioning model. -Main work: Coding+debugging+testing+analyzing.
11/24	1.5 hours: -Image captioning model. -Main work: Debugging+testing+analyzing.
12/13	3 hours: -Image captioning model. -Main work: Coding+debugging+testing+analyzing.
12/14	0.25 hours: -Stable Diffusion model. -Reading+research: <a href="https://www.youtube.com/watch?v=70H03cv57-o">https://www.youtube.com/watch?v=70H03cv57-o</a> > Installed LoRA addon for Stable Diffusion A1111. 0.5 hours: -Stable Diffusion model. -Main work: Testing+analyzing. 0.5 hours: -Stable Diffusion model. -Prep work: Dataset cleaning+preparation. 0.25 hours: -Image captioning model. -Main work: Coding+testing+analyzing.
12/15	0.25 hours: -Stable Diffusion model. -Prep work: Dataset cleaning+preparation. 0.25 hours: -Stable Diffusion model. -Main work: Testing+analyzing.
Totals:	Read+Res: 2.25 hours. Prep work: 4.75 hours. Main work: 18 hours. Total work: 25 hours.