Donkey-Net: Extending Monkey-Net to Animate Pokemon
CMSC 498L Final Project

Madhava Paliyam
Department of Computer Science
University of Maryland, College Park
College Park, MD 20740
mpaliyam@terpmail.umd.edu

Vatsal Agarwal
Department of Computer Science
University of Maryland, College Park
College Park, MD 20740
vatsalag99@gmail.com

Pauline Comising
Department of Computer Science
University of Maryland, College Park
College Park, MD 20740
acomis@terpmail.umd.edu

Abstract

Pose transfer refers to the process of accurately transferring the poses of a subject in one image to the subject of another image. Our model, Donkey-Net(Dual reconstruction moving KEYpoints), extended the work of Monkey-Net, which had used a keypoint detector, dense motion prediction network, and a motion transfer network to learn how to transfer poses. Our modified Monkey-Net architecture, spatial and channel wise attention, and perceptual loss tries to make the model more generalizable. We show that our method can slightly improve generalizablity of the monkey net model.[Siarohin et al., 2019]

1 Introduction

Image generation has been an emerging field in the world of machine learning, with Generative Adversarial Networks (GANs) and Variational Auto-Encoders (VAEs) primarily undertaking tasks of style transfer, image construction, facial expression transfer and more. Towards the more recent end of these explorations, machine learning has also been used for pose transfer. This refers to the ability to take the visual features of a source image and transfer the movement of a driving video onto it, thus creating a video prediction. Relevant methods of doing so have utilized optical flow and keypoint generation and have yielded improving results. As the field moves from singular image synthesis, as achieved by [Wang et al., 2018], to entire video reconstructions done by papers like [Siarohin et al., 2019] and [Siarohin et al., 2020], our project aims to further the realism of generated images by focusing more on maintaining original shape, size, and style after pose transfer.

Currently, Monkey-Net's ability to achieve pose transfer cleanly and realistically is highly contingent on similar visual features between the subject of the source image and driving video. Upon matching two non-similar looking GIFs, one of two things can occur: the video prediction could mold the source image's shape to exactly match the driving video's, or the video prediction could be poorly warped into a shape neither the driving or source image exhibited (se Fig. 2). Either way, both outcomes render video predictions that are near-unrecognizable to the source image. This seemed to result from Monkey-Net's self-supervised training, where source image and driving video pairs were of the same video. Through reconstruction of subjects with identical visual features, maintaining source image shape was synonymous with copying the driving video shape. Our paper looks to

Figure 1: On the top, it's apparent that the most convincing pose transfers occur with similarly shaped source and driving pairs, and differently shaped source images, such as the third, lose features such as its ears and gain features like a tail. On the bottom, the sequence is in the order of source image, driving videos, video prediction, and video deformation to illustrate unrealistic shape warping.

solve this problem with a more weakly supervised approach of manually clustering similarly moving, and therefor similarly posing, GIFs with greater shape diversity. In gathering ground truth video predictions of different animals to compare to, our model should learn to better maintain different sizes and features such as ears, wings, and tails.

Primarily working off the backbone of Monkey-Net, we sought to improve the model through a curated dataset, dual reconstruction, attention, and multi-scale loss. Dual reconstruction, which is described in more depth in section 4.2, was used to put more pressure on the model to create video predictions maintaining the driving video's pose, such that the video prediction could then be reconstructed to the driving video well enough. Attention was used through out our model, such that the generator could focus on more important parts of the reconstruction and the discriminator could better identify commonly occurring mistakes of the video predictions. Finally, multi-scale perceptual and feature matching loss was used so the model could learn a greater amount of features for reconstruction.

2 Data

In our model, we used the MGIF data set which was made publicly available by the authors of Monkey-Net. This data set is a collection of GIFs of animals each with a size 128 x 128 pixels. In our model, we used pairs of GIFs such that the poses of both the animals would have high similarity in their poses. The frames were also resampled so that both the GIFs had the same number of frames. The exact method of obtaining pairs of GIFs is explained further in section 4.

Our dataset consisted of 956 pairs of GIFs in our train set and 119 in our test set. In each sampling, we pick one pair of corresponding frames from the GIF A and GIF B, which are called driving A and driving B respectively. We also pick another frame at random from GIF A which is called source A. This is repeated three times for each pair of GIFs in our dataset during runtime.

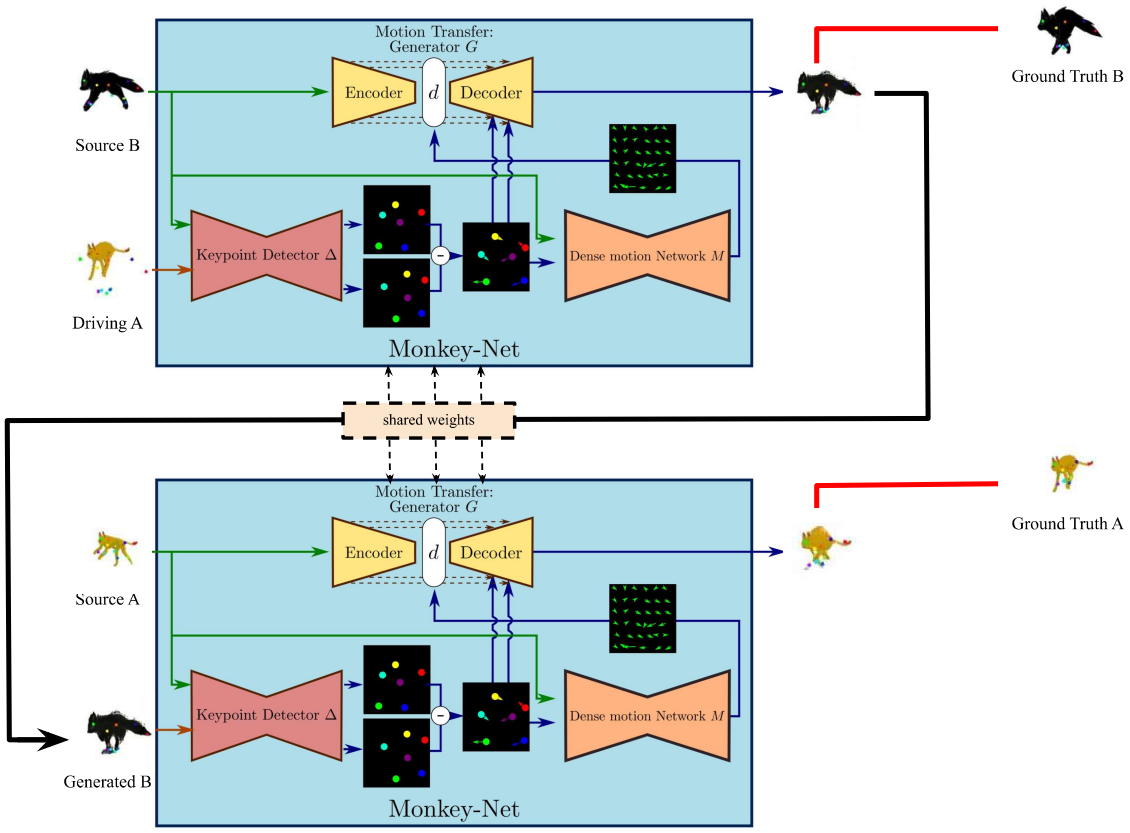

Figure 2: Donkey-Net

3 Related Work

3.1 Monkey-Net

One of the prominent methods for this include Monkey-Net, referenced earlier. As mentioned, during training time, the model takes in two different frames of a GIF and tries to transfer the pose of one to another. During inference time, a different source can be passed to Monkey-Net and it then generates images to match the driving video frame by frame.

The first module of Monkey-Net is an unsupervised keypoint detector network used to capture pose information. The network is based on a U-Net Architecture [Ronneberger et al., 2015], where the input is the source frame and the output is a series of confidence maps for 10 keypoints. A Gaussian is fit on each confidence map, and the maps of the keypoints are encoded as heatmaps. These keypoints contain information about the location of different body positions are key to to driving pose transfer. Both the keypoints of the source and target frame are extracted and then sent to the generator network.

The next set of modules make up the generator network. The first module is the motion transfer network that essentially utilizes a U-Net style architecture to capture the optical flow between the source and driving frames by taking in the difference of the keypoints at the source and target frame and the original source image. This module then warps the source image based on the optical flow. This is then passed into the image generator module which also follows a U-Net architecture and then creates the final reconstruction.

Finally, loss is calculated as the discriminator network tries to differentiate between the actual driving image, and the generated driving image. This is done by concatenating the keypoints of the driving image to both the real driving image and the generated driving image and applying the least squares GAN loss. They also use a feature matching loss in the discriminator to encourage similar video prediction and driving video feature representation.

3.2 Attention

Attention can help a neural network focus on a specific part of the input. In our case, we use attention to help the generator and discriminator focus on the most important features of the image to reconstruct it. There are many ways of adding attention to a neural network and in this section we will explain the methods that we took into account in our research.

3.2.1 Convolutional Block Attention Modules (CBAM)

CBAM is introduced in [Woo et al., 2018] where the authors use both channel-wise attention and spatial attention on each feature map. Following the work of [Hu et al., 2017], the channel-wise attention module first applies average and max pooling on the feature map and concatenates these two vectors and passes them to a fully-connected layer with sigmoid activation to obtain the attention weights. The channel-attention weights are multiplied upon the original feature map and then fed to the spatial attention. For the spatial attention module, an average pooling and max pooling on the channel-refined feature map is concatenated and fed through a convolutional layer. The attention maps are added element-wise with the original feature map and passed further downstream in the network.

3.2.2 Bottleneck Attention Modules (BAM)

BAM [Park et al., 2018] computes attention similarly to CBAM in both the channel and spatial dimensions with a few differences. Specifically, the channel attention and spatial attention are computed independently rather than sequentially. Furthermore, the spatial attention is generated by passing the pooled spatial tensor through a series of dilated convolutions rather than a single convolutional layer. With regards to integration, these modules are placed at the bottlenecks of a standard network where downsampling occurs. The authors show that adding attention at bottlenecks improves classification and detection tasks.

3.2.3 Efficient Channel Attention (ECA)

In an effort to reduce the number of parameters in pooling based methods such as [Hu et al., 2017], [Wang et al., 2019] proposes an efficient channel attention module which adaptively selects the size of a 1D convolution based on the number of channels in the input feature map. They show that this reduces the number of parameters in the model and reduces dimensionality reduction while effectively capturing cross-channel interactions and improving performance.

4 Approach

4.1 Data Preprocessing

Our approach aimed to maintain shape and features, such as tails and ears, after pose transfer occurred. Training this is only possible with the aforementioned GIF pairings of differently shaped animals, and we experimented with a few methods to achieve usable pairs.

4.1.1 VGG-11 and Affine Similarity Clusters

Similar movement clusters were first created with the assumption that similarities in the first frame would be representative of entire videos. To group GIFs, cosine similarity using VGG-11 [Simonyan and Zisserman, 2014] fed feature vectors and affine similarity were used to try and cluster on visual features, primarily size and shape, and pose, respectively. With the aim of grouping similar animals in close starting positions, GIFs with similar movement could potentially be clustered. VGG-11 similarities were calculated through the difference of feature vectors outputted by a pretrained model that took in the original image, the grey-scaled image, and edges. Affine similarity was able to estimate pose similarity through a transformation matrix's ability to transform one GIF's keypoints, X, into the other's, Y. With a least square solution A to such transformation, the difference between the resulting keypoints and the expected keypoints measured whether first frames had close to the same pose or not.

$$A \cdot X = Y$$

$$A \cdot X = \hat{Y}$$
$$res = \hat{Y} - \hat{Y}$$

Similarity matrices were created for both, and the intersection of the best scoring VGG-11 and affine similarity sets was found for each GIF, creating the subset of similar GIFs to resample.

Many issues with this process led us to ultimately switch to manual clustering. Primarily, VGG-11 similarity would favor similarly colored animals, which weren't indicative of movement, and the least squared solution for keypoint transformations often created very close keypoints to the expected one, even with drastically different pose pairs. As a result, the job of resampling wasn't able to create similar enough GIFs to train on, and we switched to various manual clustering endeavors.

4.1.2 Manual, Movement Clusters

Upon attempting to better create the similar GIF sets, we decided grouping directly on movements and paying special attention to animal type was more effective. GIFs underwent a preliminary split on direction, left or right. Afterwards, groups of gallopers, walkers, and more were created, while more unique movers were put together as well. This simpler, albeit more time consuming, process resulted in better movement clustering.

4.1.3 Frame Resampling

With clusters ensuring overall similar movement between GIF pairs, frame resampling aimed to make frame-by-frame poses as close as possible. For a pair of GIF A and GIF B in a cluster, frames of GIF B were chosen and reordered to match those of GIF A, in poses and in length, and vise versa (see 3). Affine similarity was used to choose the most similarly posed frames for resampling.

Thresholds on mean affine difference and frame uniqueness were used to further narrow down our final dataset and filter out pairs that could not be resampled to adequately similar GIFs. The affine difference of each frame pair was averaged over the GIF's length, allowing for an average residual to be calculated with:

$$avgres = \frac{1}{nframes} \sum_{frames} (Y - \hat{Y})^2$$

Another mark of poor GIF pairing was repeatedly choosing the same frame as the resampling. As mentioned, we created a unique frame threshold to deal with this issue, and settled on a threshold of 0.25 frame repeats. Finally, the interplay of affine difference and unique frame count was weighted with alpha, such that overall resampling difference was calculated by mean affine residual - alpha * unique frames. Our final alpha and difference thresholds came out to 0.8 and 0.14, respectively.

4.2 Dual Reconstruction

The next part of our approach focuses on loss calculation and reconstruction. Monkey-Net operates with 2 frames for each iteration, a driving frame whose pose is transferred onto a source frame. With access to two GIFs, A and B, for each iteration, our model extracts 4 frames: one driving frame with the same index, and therefore same pose, from each A and B, a random source frame from A, and a random source frame from B (see Fig. **??**). Dual reconstruction occurs as driving A's pose is transferred onto source B, and that prediction transfers its pose onto source A, resulting in prediction B and prediction A, respectively. Loss is calculated by comparing driving A and prediction A, and driving B and prediction B.

4.3 Channel and Spatial Attention

To further achieve greater accuracy, attention mechanisms were added throughout the network, with an eye for integration ease and efficiency. As not to drastically change Monkey-Net's preexisting architecture, attention modules, CBAM and BAM, were added seamlessly between blocks of the Dense Motion Module, Generator, and Discriminator. Following [Oktay et al., 2018], we add CBAM modules to the U-Net in between the upscaling in the decoder module. For the discriminator, we add a BAM module between each downblock where important features are reinforced in distinguishing fake versus real images.

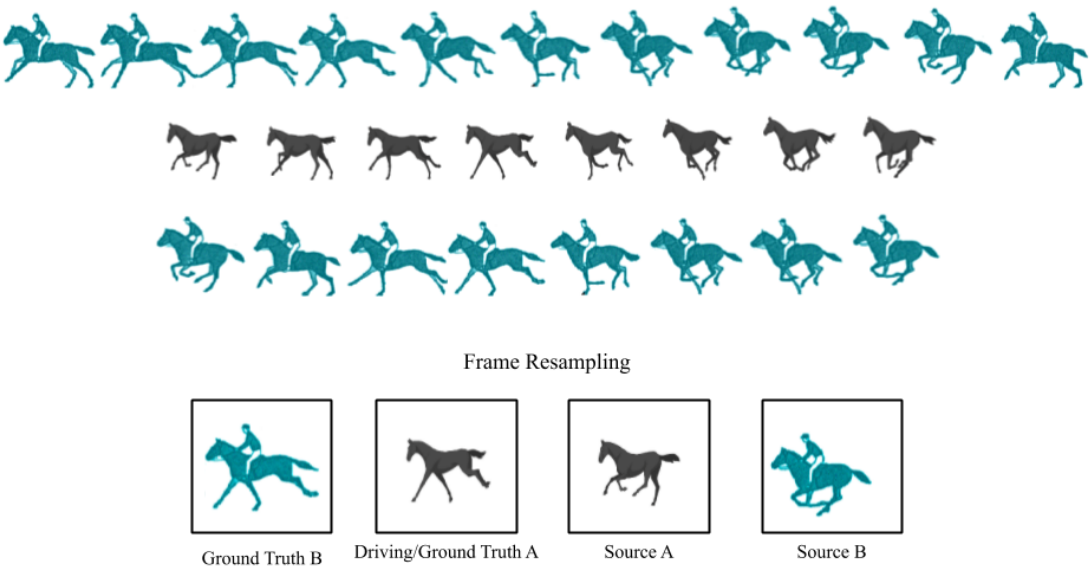

Figure 3: The top row is the GIF whose frames will be resampled to the second row's, resulting in the third row GIF frames. Underneath are an example of 4 frames that our network trains on.

Both modules look at channel and spatial attention to better inform both what to look for and where. While CBAM's and BAM's spatial attention mechanisms were maintained, the fully connected layers within their channel attention mechanisms posed complications when given different sized feature vectors each block. Within both the generator and discriminator, the layers outputted different feature vector sizes, so instead ECA was used to replace the original channel attention mechanisms. With the addition of attention, we hoped that our generator would be able to better focus on image reconstruction tricks for the most convincing predictions and that the discriminator would also be better at distinguishing fakes, although trade offs such as over fitting are explored more later.

4.4 Multi-Scale Perceptual and Feature Matching Loss

The final changes we tried were multi-scale loss additions in hopes of increasing learned global and local features and accuracy. Multi-scale feature matching loss was simply achieved, as the singular-scale loss already existed and was just calculated for an additional 2 down samplings of the original feature vectors. For both multi-scale losses, our 3 scales were 1, 0.75, and 0.5. Perceptual loss uses a pretrained network and takes in two images and determines whether aspects of spatial structure color, texture, and more are similar. In contrast, this loss required a little more work, and was added to focus on style transfer and further take advantage of our ground truth video predictions. In emphasizing perceptual loss between the video prediction and our ground truth images, the differences that could not be learned through Monkey-Net's self-supervised dataset were under close scrutiny within our model. Our perceptual loss was calculated through the integration of a pretrained VGG-19 model's feature extraction. Unfortunately, some of these changes didn't operate as expected and will be discussed later.

5 Experiments and Results

Due to constraints on time we were not able to do a full ablation study, however in this section we will discuss the results that we obtained from our experiments. All of our models were written using Pytorch [Paszke, 2017] and run on Google Cloud GPUs. We also used the pretrained model trained for 750 epochs on the original MGIF dataset made available by the authors of Monkey-Net as our starting point. Each model was trained for 100 epochs. For the first two models run without attention, we used a batch size of 10, however due to memory constraints for the next models, we decreased the

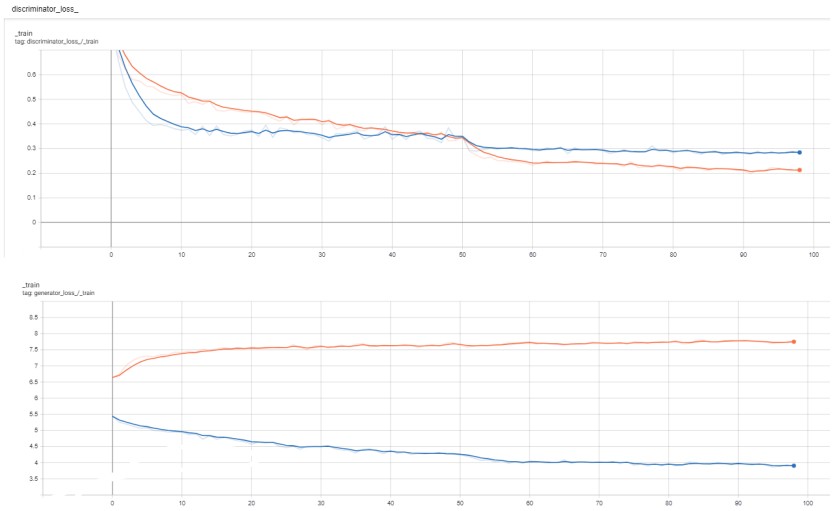

Figure 4: Discriminator GAN loss and Generator GAN loss while using ground truth keypoints (orange) and approximate keypoints (blue)

batch size to 8. In addition, due to time constraints we were unable to extensively test on our test dataset.

5.1 Keypoint Selection

The discriminator uses keypoints and an image to determine if the image is real or fake. We tried two experiments to decide which pairs of keypoints to send to the discriminator. First, we sent the keypoints of the ground truth image, the loss for this is shown in blue in Fig. 4. We then tried running the same model instead sending the approximated key points from the driving image, the loss for this is shown in orange in Fig. 4. We see that the images (add images) show a clear difference between the two methods. Images not using the ground truth keypoints had abnormal limbs, while using the ground truth meant that the original features were preserved 4. The loss also shows that using the ground truth meant better convergence for the generator loss. This could be due to the fact that using the approximated keypoints resulted in the model trying to warp the prediction to try and create image features from the driving video.

5.2 Attention and Multi-Scale loss

We experimented with using channel wise and spatial loss in our model. We also calculated feature matching loss over multiple scales for our discriminator network. Results with and without using attention and multi-scale loss are shown in Figure (add images). While using only attention, we see that there is some scraping and fuzziness on the generated images, while using both attention and multi-scale loss, we see more accurate results.

5.3 VGG-19 Multi-Scale Perceptual Loss

We experimented with adding a pretrained VGG-19 network to calculate perceptual loss at different scales. The loss was calculated by taking the difference of the feature vectors produced by VGG-19 when the generated image and ground truth image was used as input. This was repeated two more times with .75 and .5 scale factor to help the generator keep features across different sized images. Interestingly, we saw that by the 200th iteration, the generated images were faded and the discriminator loss was very high compared to our previous trials. This most likely happened because the generator figured out that by making a faded image the discriminator is not able to tell the difference between real and fake image. This could potentially be happening because the perceptual loss is a big portion of our total loss. An example of the output is shown in Figure 6.

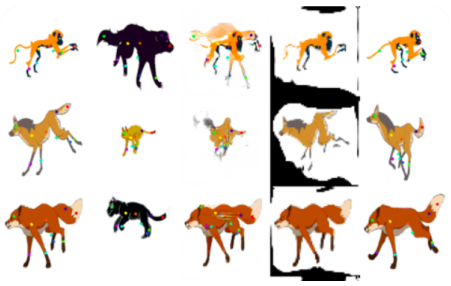

Figure 5: Results from using driving video and ground truth keypoints, revealing poor shape and size maintenance.

Figure 6: Results from perceptual loss. The faded appearance can be seen in frame 3.

5.4 Video Translation Results

As per our title, we tested out program on a few different Pokemon. Similar to the Monkey-Net, we saw some examples which we did not translate as well, but some results were promising. These examples can be seen in Fig. 7. We see that generally, the model is able to better maintain the shapes of the source image compared to the original model which loses some of the image details such as foot shape or the head. However, a pertinent issue which arose was that the model did not move the source images as well as the original Monkey-Net model. This could be attributed to our training method of clustering GIFs and resampling as this would probably have led to some mismatched frames and confused the model.

6 Conclusion and Discussion

By training on clustered, preprocessed data and carefully choosing the most effective loss mechanisms, our model was able to effectively learn size and shape maintenance. Donkey-Net's video predictions were able to maintain unique source image features such as horse back riders and wings, unlike previous models. This success was rooted in multiple iterations of narrowing down our dataset and sending ground truth keypoints to our discriminator loss as discussed in Section 5.1. In the absence of a proper ablation testing, the exact effects of attention and multi-scale loss were unclear, but did not poorly affect our model.

While not too intellectually intensive, the issue of curating our dataset posed a big problem as it demanded a lot of attention to detail. Inventory of early training results often revealed misleading ground truth datapoints. As a result, concurrent to architecture and loss changes, our dataset was also changing between experiments. This issue pointed to our resampling code and formula's inability to properly weed out poor GIF clustering, challenging the assumption that a low affine difference was an adequate measure of pose similarity. Our resampling thresholds were only effective when used alongside detail-oriented, manual clustering.

Our other problems and currently unexplainable findings center on reconstruction and the question of how much information should be inputted to our generator. One previous issue led to a form of mode collapse where the generator was able to fool the discriminator by simply copying the source image without any pose transfer at all. Possibilities of sending down sampling or feature representations of our source image were proposed to stop copying, but this issue was ultimately solved through disabling replacement when choosing our set of 4 datapoint frames. In decreasing the probability of the source image matching the ground truth image, the generator no longer learned to perfectly recreate the source.

As for currently unexplainable findings, results show that the second reconstruction has consistently out performed the first, showing more signs of potential overfitting. Despite presumambly being at a disadvantage with a reconstruction as its source image, the final video predictions have come out very close to our ground truth, regardless of how well the first reconstruction goes. Upon investigation of both Generators' input, neither one should be receiving more information, and, as far as we can tell, only the video prediction was sent in.

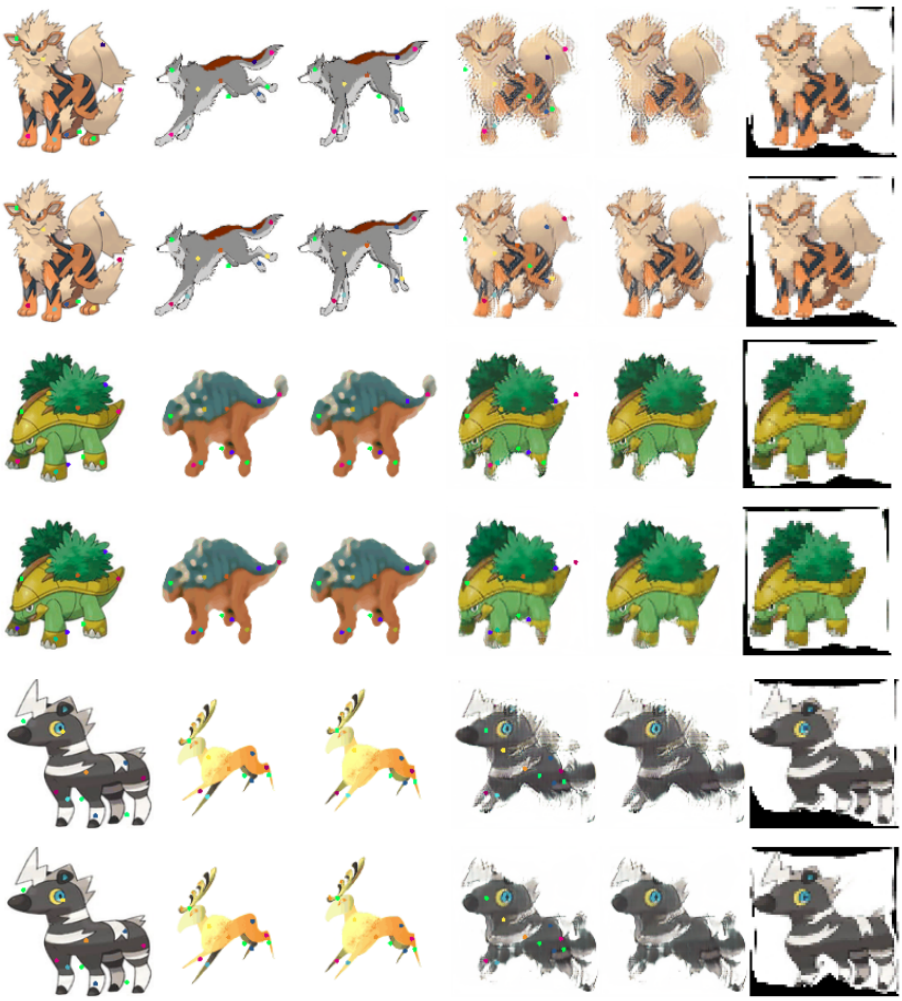

Figure 7: Top: Attention only, Bottom: Multi-scale.

Figure 8: Same source and driving pairs ran through Monkey-Net

7 Future Work

Moving forward, we'd like to explore hyperparameter fine tuning, ablation testing, issues of overfitting, and different dataset application. Given another couple of weeks, the first three goals could be achieved. In focusing more on the architecture and loss of our model within a shorter time period, training on different hyperparameters such as batch size, learning rate, and epochs, and singling out different changes was not as feasible. Utilizing better understandings of those decisions and combinations could then serve as solutions for overfitting. Exploring ways to ensure greater generalizability would focus on the balance of ground truth videos' and driving vidoes' influence on prediction loss. More specifically, making sure subtle pose differences between GIF pairs did not result in poor pose matching between driving a video prediction is vital. After finishing both of these, we'd aim to apply this better form of pose transfer to different datasets such as dancing people and see if different encoder decoder architectures could improve motion transfer on these new videos.

References

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017. URL `http://arxiv.org/abs/1709.01507`.

Ozan Oktay, Jo Schlemper, Loïc Le Folgoc, Matthew C. H. Lee, Mattias P. Heinrich, Kazunari Misawa, Kensaku Mori, Steven G. McDonagh, Nils Y. Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas. *CoRR*, abs/1804.03999, 2018. URL `http://arxiv.org/abs/1804.03999`.

Jongchan Park, Sanghyun Woo, Joon-Young Lee, and In So Kweon. BAM: bottleneck attention module. *CoRR*, abs/1807.06514, 2018. URL `http://arxiv.org/abs/1807.06514`.

A. Paszke, et al. Automatic differentiation in pytorch. In *NeurIPS Autodiff Workshop*, 2017.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2377–2386, 2019.

Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks, 2019.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. *Lecture Notes in Computer Science*, page 3–19, 2018. ISSN 1611-3349. doi: 10.1007/978-3-030-01234-2_1. URL `http://dx.doi.org/10.1007/978-3-030-01234-2_1`.