Implementing In-Context Learning In Diffusion for Key Point Generation

Author: Jin Schofield
Mentors: Professor Pietro Perona, Rogério Guimarães

Abstract. Latent diffusion model training is a computationally heavy process that prevents the flexible instilling of expert knowledge after the process' completion. In-context learning is a form of learning in transformer-based language models where information can be taught to the model through an input prompt without having to re-train weights. An important problem in computer vision is that of key point generation, often applied in tasks such as pose estimation. This research applies a process analogous to in-context learning to unsupervised key point generation using latent diffusion to allow key point generation to benefit from added knowledge without having to re-train the model. This is accomplished by appending an initially randomized embedding (which is later optimized by a loss function) to the original embedding conventionally used in cross-attention during latent diffusion. This randomized embedding is trained on a small new dataset of five images paired with attention maps whose maxima are desired key points. After optimization, this new embedding guides the cross-attention mechanism to create attention maps concentrated onto the left elbow in test images in 26.25% of cases, as opposed to near zero originally. Increasing the training dataset size from 5 to 20 did not improve accuracy. Accuracy for identifying the head was 12.00%, and that for identifying the left elbow was 0.00%. Key points are the coordinates most impacted by attention in an image. The embedding is created using a loss function minimizing mean squared error between pixels of predicted and ground truth desired attention maps, optimizing localization of Gaussians in the attention maps, and maximizing equivariance of the key points between transformed versions of the same image. This research allows us to understand and improve how diffusion models create robust and modular abstract representations that meaningfully and efficiently encode semantic information in images.

Introduction

In computer vision, an efficient method exists to identify key points — parts of an image that would appear important to humans as well — using unsupervised machine learning that does not require re-training of diffusion models or labeling of large new datasets. However, if a person wished to utilize expert data and design the key points to occur in a specific area, this would not be possible in an efficient manner. Before delving into the research of this paper, which explores how to implement efficient specific key point generation, some background information will be discussed.

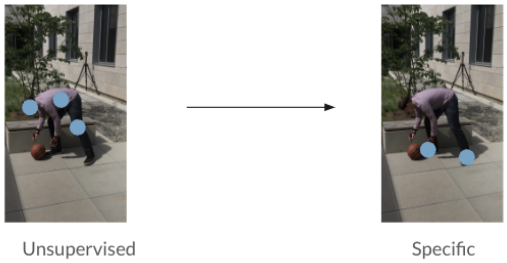


Figure 1: The task that this research aims to accomplish: shifting key point generation to specific, desired locations

Background

Diffusion is the process by which Gaussian noise is gradually added to images in a dataset until the original image is converted entirely to noise. A function learns the noise that is added to the images. Then, an image that is initially entirely noise can use this function to remove noise (denoise) until the remaining pixels resemble an image that could have been sampled from the original dataset.

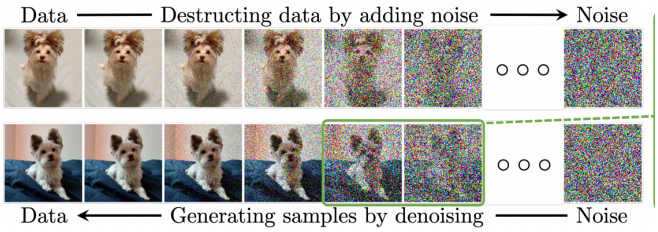


Figure 2: The process of diffusion (Introduction to diffusion models for machine learning)

The process of diffusion can be quite inefficient when applied to the actual pixels of an image. Latent diffusion can increase the efficiency of diffusion. The images from the original dataset are sent through an encoder that distills the image into a latent representation, which is an encoding of the image's high-level features. Diffusion is then run on this latent representation. When an image is denoised, it is run through a decoder, which converts the denoised image encoding into an image in the pixel space. Running diffusion in the latent space rather than the pixel space preserves accuracy while increasing efficiency.

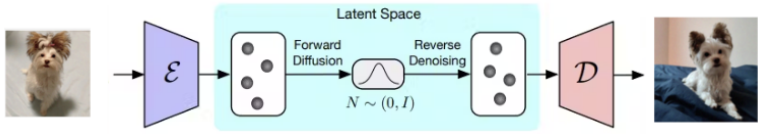


Figure 3: The process of latent diffusion. (Cong et al.)

Cross-attention is a mechanism that allows for images to be denoised with an added condition of an embedding. For example, in popular diffusion models such as Midjourney, the text prompt that a user gives the diffusion model is converted into tokens, which are represented as a numerical embedding. The embeddings are converted into key-value pairs. The image that needs to be denoised is converted into queries. The queries and key-value pairs are matched and guide the removal of noise from the image.

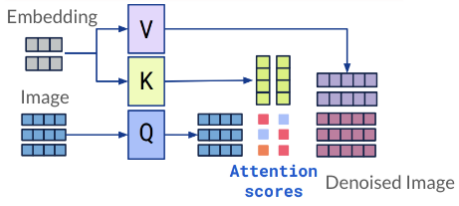


Figure 4: A diagram of the cross-attention mechanism (Kosar)

Attention maps can be printed during this process. They can also be displayed showing how each token impacted attention.

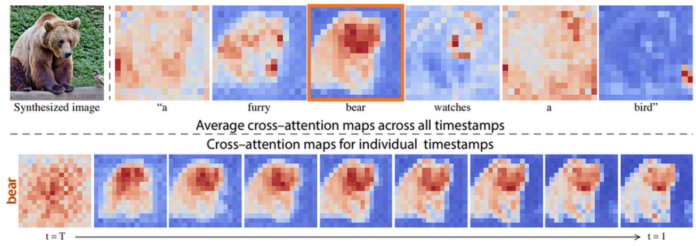


Figure 5: (top) Attention by token, (bottom) Attention by timestamp

Key points can be generated with the help of the cross-attention mechanism. In one particular paper, a randomized embedding is used in cross-attention (Hedlin et al.). A loss function is used to optimize the embedding that conditions for the attention maps to form single-mode Gaussians that land on equivariant points regardless of image transformations. When the embedding is optimized, the attention map formed during the diffusion contains single-mode Gaussians (one for each key point). The maximum coordinate of these Gaussians are considered key points. Note in figure 5 how these key points are generated in an unsupervised manner and, although equivariant, do not generate on a particular point specified by the user of the algorithm.

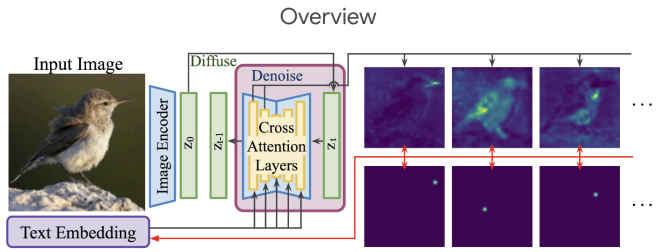


Figure 6: The architecture from "Unsupervised key points from Pretrained Diffusion Models" (Hedlin et al.)

The images below are some samples of the unsupervised key point generation from this model. The number of key points in the sample generations vary from 1 to 10. Due to key points having some distance enforced between them, it seems that the model works better and more consistently when generating ten key points at a time rather than less.

Figure 7: Example generations from the unsupervised key point generation model from "Unsupervised key points from Pretrained Diffusion Models" (Hedlin et al.)

Another interesting paper that inspires this research is "An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion" (Gal et al.). In this paper, a standard diffusion model generates images of a particular theme, such as paintings. A new idea not trained into the diffusion model, such as that of a cross-legged statue, is converted into an embedding. This embedding is appended to other embeddings used by the diffusion model. The appending of this new embedding representing this new idea allows the original diffusion model to generate an image of its original theme with the new idea incorporated into it, i.e. an image of a painting of a cross-legged statue. This would not have been possible without the appending of the new embedding unless the diffusion model was trained again with many examples of this new idea, which would be inefficient. This paper demonstrates how appending embeddings to other embeddings can augment the output of a diffusion model without having to re-train the diffusion model.

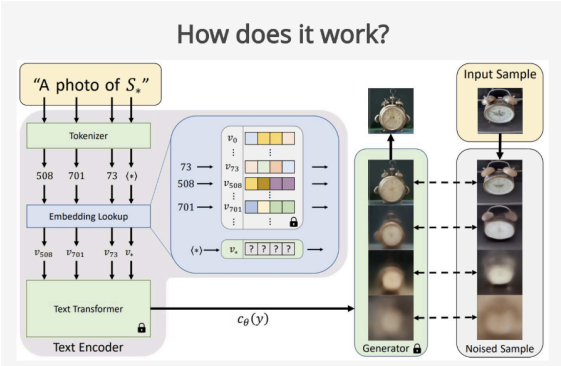


Figure 8: The mechanism behind textual inversion (Gal et al.).

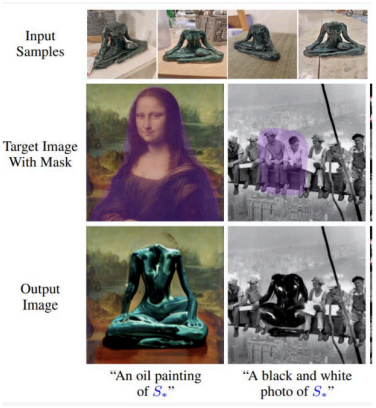


Figure 9: An example of image generations using textual inversion (Gal et al.).

A final important concept that inspires this research is that of in-context learning. In-context learning is a notion in natural language processing of improving a model's accuracy without retraining the model's weights. In NLP specifically, this occurs by adding context (such as input-output pairs) into input prompts that are tokenized and given to the model. This research aims to implement in-context learning — that is improve a model's performance without re-training by providing more context — in the realm of computer vision and diffusion.

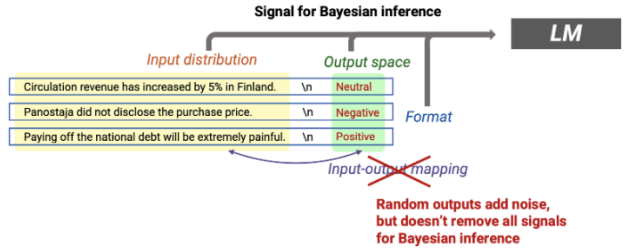


Figure 10: An example of how in-context learning works in natural language processing (Xie and Min)

Problem

This research aims to allow for the incorporation of expert knowledge, or the increase in flexibility of usage, of diffusion-based unsupervised key point generation. Specifically, we aim to cause key points to be generated in specific places in an image specified by a user. We aim to do this without retraining diffusion model weights and without collecting a large dataset. We will specifically be aiming for the key point to be on the left foot, head, and elbow. Solving this problem would be useful for users of diffusion models who want to apply key point generation efficiently for a specific use case where context might be important (i.e. sports pose estimation). To clarify, a key point is the coordinate pair of maximum attention in an attention map. At the end of this paper, we will demonstrate how we were able to have the point land on a desired key point 26.5% of the time when, without using this method, it landed on the intended key point coordinate very rarely (near zero).

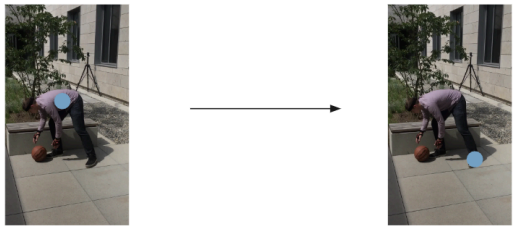


Figure 11: We intend to shift the key point to a specific desired area rather than a vaguely important key point auto-generated by the unsupervised key point generation method (Hedlin et al.).

Methods

The dataset I use is from the Max Planck Institute (3D poses in the wild dataset). I use a training dataset of 5 images for some iterations and a training dataset of 20 images for other iterations. The test dataset is 10 images.

The Three Layers of Computation

There are three layers of computation in my approach. There are two frozen layers, the diffusion model itself, and the embedding generation from the unsupervised key point generation paper mentioned in the background (Hedlin et al.). This second layer uses a loss function that optimizes for the creation of single-mode Gaussians in the attention maps (localization) and for key points to land in the same place regardless of image transformations (equivariance), My contribution is adding a third layer, an extra embedding appended to the embedding in the second layer that uses a loss function that minimizes mean squared error between all the pixels in a ground truth attention map and the predicted attention map for training images.

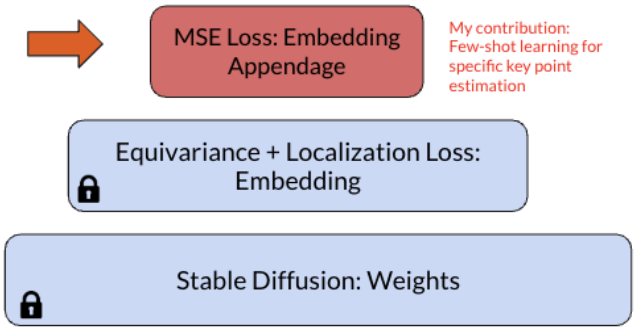


Figure 12: The three layers of my approach
‘

The following is a diagram of the final approach. I append an embedding to the original embedding from the unsupervised key point generation method (Hedlin et al.).

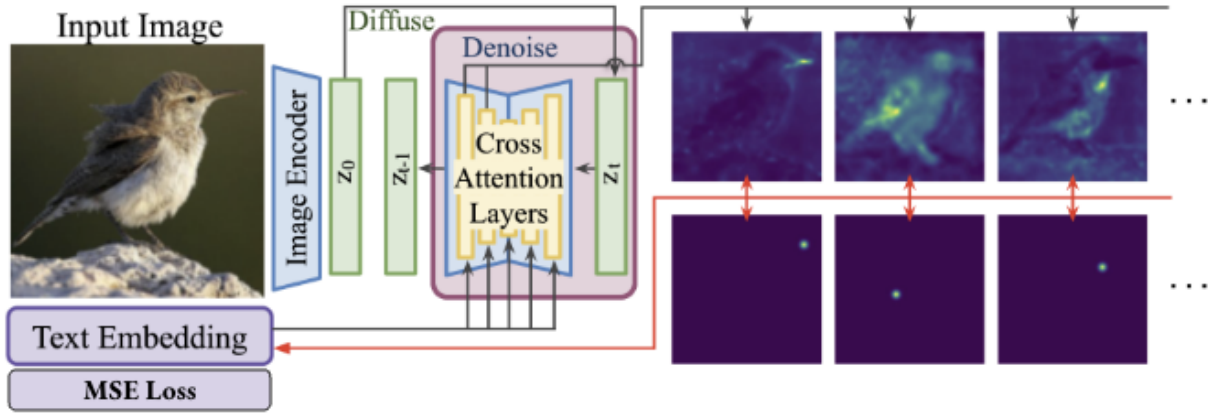


Figure 13: The architecture of my method

Metrics of Success

Overall, I will be aiming to maximize the proportion of predicted key points that land within a specific pixel range of my desired key point.

While working towards this ultimate metric, I use two metrics to guide my hyperparameter optimization. Firstly, I calculate the mean squared error between the attention in the pixels of the ground truth attention maps and attention in the pixels of the predicted attention maps. These ground truth attention maps are generated so that the pixel with the maximum attention is where the desired key point is located.

The second metric is looking for semantic meaning in the error. For example, in figure 14, the left photo shows a lack of consistency in the direction of the error, whereas the right photo's key points fixate on the head (this generation is intending to place the key point on the foot). This indicates there may be a different hyperparameter to be optimized between these two types of errors. For example, the left image would benefit from weighting equivariance more in the loss function while the right would not.

Figure 14: Examples of semantically meaningful and semantically meaningless error

Failed Approaches

1. MSE of Only Maximum

In this approach, the loss function for the third layer of my approach minimized the mean squared error between the desired key point coordinate and the predicted key point coordinate. This did not improve accuracy and was not used because it did not reflect the nature of attention maps, which are integral to how diffusion functions. It thus did not provide a differentiable function along which the loss could improve.

2. Using CLIP and YOLO

A second approach was to use YOLO to create a bounding box around predicted key points (Redmon et al.). Then, the portion of the image inside the bounding box is processed by CLIP and converted into an embedding (Ratford et al.). The MSE between the embedding of the predicted key point and an embedding generated from the area of the image of the desired key point is used for the loss function. This did not improve accuracy. It was not used because CLIP is trained on entire objects, not parts of objects, and thus could not properly create embeddings for parts of the human body from my dataset. Additionally, this method does not provide a differentiable, smooth loss map to traverse. A hybrid version of the MSE of maximum point approach and this CLIP and YOLO approach was also tested, but did not improve accuracy.

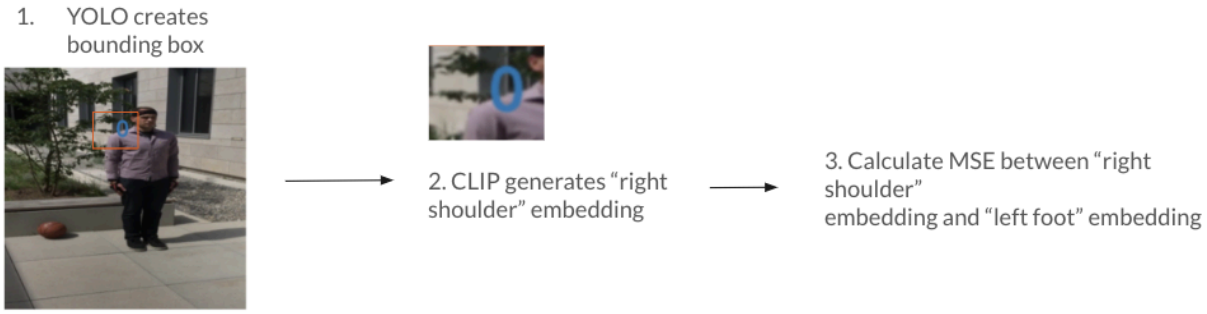


Figure 15: A diagram of the failed approach using CLIP and YOLO

The final hyperparameters were the following. How these hyperparameters were optimized and what they are is explained in the discussion section of this paper. Ground truth maps utilized a Gaussian distribution with a standard deviation of 0.05 times each dimension of the training image. The third layer incorporated equivariance and localization into the loss function in addition to MSE of attention maps, with a relative weight of 1 localization, 1 equivariance, and 5 MSE. The relative size of the appended embedding in the third layer that is appended onto the second layer embedding is 0.2. 1 augmentation step and 100 optimization steps were used. All hyperparameters were optimized by comparing to results when optimizing for the left foot.

Results

When training on only 5 training images and optimizing for the left foot, 26.25% of key points landed within 300 pixels of the desired key point in a meaningful direction (or the yellow area in figure 16) with a standard deviation of 18.06%. 17.50% of key points landed within 40 pixels of the desired key point in any direction (or the orange area in figure 16) with a standard deviation of 15.86%.

Figure 16: Shaded Areas for Results of Left Foot Optimization using 5 Training Images

When training on 20 training images and optimizing for the left foot, 21.00% of key points landed within 300 pixels of the desired key point in a meaningful direction (or the yellow area in figure 16) with a standard deviation of 27.89%. 14.00% of key points landed within 40 pixels of the desired key point in any direction (or the orange area in figure 16) with a standard deviation of 15.72%.

When training on 20 training images and optimizing for the point to land on the head, 12.00% of key points landed within 200 pixels of the desired key point in a meaningful direction (or the yellow area in figure 17) with a standard deviation of 16.85%. 9.00% of key points landed within 40 pixels of the desired key point in any direction (or the orange area in figure 17) with a standard deviation of 15.94%.

Figure 17: Shaded Areas for Results of Optimization for Head key point using 20 Training Images

When traicaltechning on 20 training images and optimizing for the point to land on the left elbow, 0.00% of key points landed within 40 pixels of the desired key point in any direction (or the orange area in figure 18).

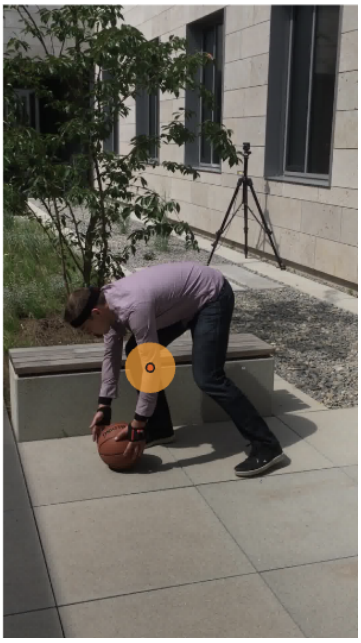


Figure 18: Shaded Areas for Results of Optimization for Left Elbow key point using 20 Training Images

Qualitative findings include how, when optimizing for a left foot key point and the key point is not landing where desired, often the point lands on easily identifiable objects with distinct colors and shapes such as the head or basketball in the image.

Figure 19: Key point landing on head consistently despite desired key point being left foot

Additionally, accuracy seemed to rely heavily on the embedding. For example, for any given run using one embedding, it was more likely for 0/10 or 3/10 key points to land in the correct place, not 1/10 or 2/10. The correct hits come in clusters.

Discussion

Multiple hyperparameters were optimized.

1. Type of Ground Truth Attention Map

I considered both the Gaussian and Cauchy distribution to form the ground truth attention maps. I considered Cauchy because it has thicker tails which would provide more learning signal to attention at the tails, allowing the maximum point to move closer to the desired maximum. It also has a steeper approach to the maximum within two standard deviations of the center, which allows for the predicted point to be more concentrated in the center. I did not end up using the Cauchy distribution because the localization aspect of the loss function in the second layer optimizes for Gaussian-shaped distributions, and I wanted the elements of the loss function to be consistent. Using Cauchy also did not improve accuracy.

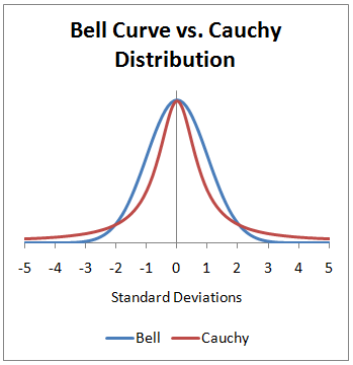


Figure 20: Gaussian vs Cauchy Distribution

2. Standard Deviation of Gaussian

The ideal standard deviation for the Gaussian in the ground truth attention maps was experimentally determined to be 0.05 of the size of each dimension of the image.

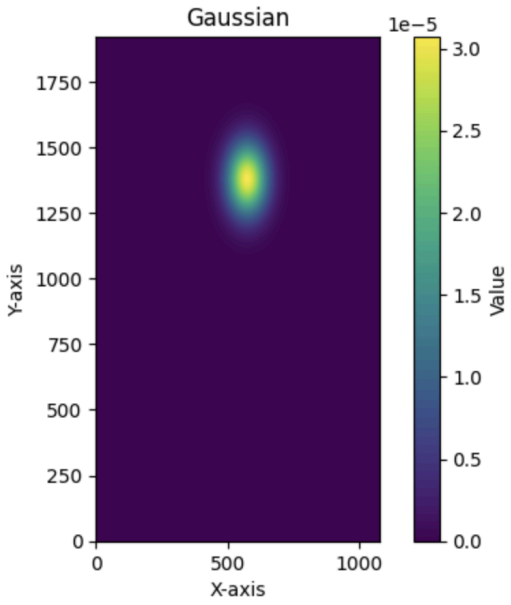


Figure 20: An example ground truth attention map

3. Whether to Combine Layer 2 and Layer 3

Experimentally, it was more effective to add equivariance and localization calculations to the loss function for the third layer that originally only calculated MSE of attention maps. This is likely because the equivariance and localization calculations acted as regularizers. They reduced the search space and thus minimized overfitting and avoided the key points landing on local minima.

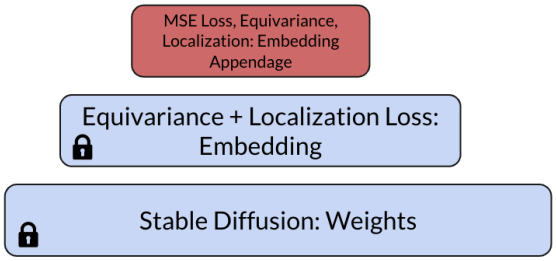


Figure 20: Diagram of updated layers

4. Weights of Layer 3

Experimentally, the relative weighting of localization, equivariance, and attention map MSE in the embedding appendage was optimized to be 1 localization, 1 equivariance, and 5 MSE.

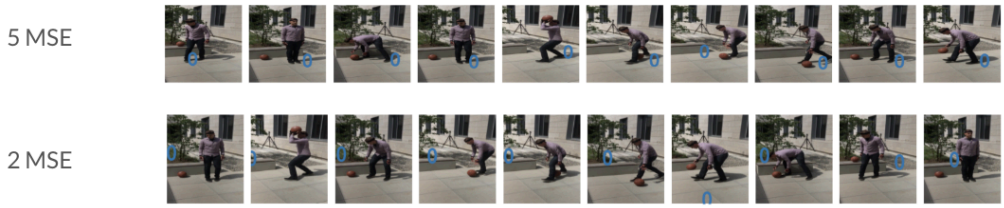


Figure 21: Examples of key point generation with varying relative weights of third layer loss function

5. Size of Appended Embedding

Experimentally, the size of the appended embedding (that is, the size of the third layer embedding which is appended to the second layer embedding) was optimized at 0.2 times the size of the second layer embedding.

Figure 22: Examples of key point generation with varying appended embedding sizes

6. Number of Augmentation Steps

Augmentation steps are the number of times that the attention is re-run with the same embedding but with different transformations (rotations, scaling, translations, etc…) to improve robustness. The coordinate locations of the key points are averaged across all augmentation steps. Although the original unsupervised key point generation paper used 10 augmentation steps, 1 was found to be experimentally optimal. At this point, the model identified the left foot around ~20% and points were either very close to the correct point or very far (only rarely moderately far). Thus, I believe 1 step was optimal because averaging the outcomes for ten photos would often cause the average for each of the ten photos to never be on the left foot because the correct points would be averaged with the incorrect points.

7. Number of Optimization Steps

Experimentally, the optimal amount of optimization steps was 100.

Figure 23: Examples of key point generation with varying numbers of optimization steps

It seems that increasing the number of training images did not meaningfully improve accuracy. This is likely due to how the increase from 5 to 20 images is not large in comparison to datasets used to train the diffusion model.

The accuracy dropped when optimizing for the head. This is likely because hyperparameters might be overfit for optimizing for the left foot since they were optimized while running experiments for the left foot key point optimization. The accuracy for the identification of the left elbow was very low. This signifies that this method works more effectively on key points that are visibly different from the rest of the image, such as having a contrasting color.

There was also some interesting behavior likely due to randomness, although none was replicable. Firstly, with one optimization step, there was an instance where the left foot was identified 100% of the time.

Figure 24: Left foot optimization with 1 optimization step with 100% accuracy (not replicable)

Additionally, even at suboptimal hyperparameters (e.g. third layer embedding being the same size as second layer embedding), the key point was capable of landing on the head consistently or the basketball consistently. This result, as seen in the following photo, occurred when optimizing for the key point to land on the left foot.

Figure 25: Left foot optimization whose key point lands on head many times

Conclusion

In conclusion, using a supervised method with 5 input images, in-context learning was able to increase the ability of the key point generation to target the left foot from near zero to 26.25%. More identifiable body parts as key points are better suited for this method. Increasing the size of the training dataset from 5 to 20 does not increase accuracy. Applying the same method to locating the head gave 12.00% accuracy, and doing so for the left elbow gave 0.00% accuracy.

Implications

This work signifies that it is possible to optimize key point generation for specific semantic locations and use cases while maintaining the efficiency of unsupervised key point generation. This can be useful in pose estimation. It also broadly shows how in-context learning can be implemented in computer vision.

Next Steps

I plan to pursue several next steps.

Firstly, I plan to train the original embedding (that of the second layer) while aiming for ten key points (which is a controllable parameter), which seems to make the range of key points produced more stable, equivariant, and semantically salient. Then, I will proceed as normal and train the third layer looking for one key point. As of this paper's writing, I trained the second layer embedding looking for one key point.

I also intend to train the original embedding on unlabelled images from the same dataset used for the third layer embedding to take advantage of semi-supervised learning.

Thirdly, I plan to implement the object similarity metric as a more precise measure for accuracy and perhaps incorporate it into the loss function.

I will also try to increase accuracy by tuning hyperparameters while testing results on various different desired key points rather than only one (e.g. head, left elbow, etc…). My hyperparameters as of this paper's writing were trained while only looking at results of left foot key point optimization.

I also intend to experiment with larger training datasets, more varied training datasets, and while aiming to detect multiple specific key points at a time.

In the far future, I hope to explore ways to better represent semantic meaning by, for example, comparing embeddings in some way that forms a differentiable loss function.

References

3D poses in the wild dataset. 3DPW | Real Virtual Humans. (n.d.). https://virtualhumans.mpi-inf.mpg.de/3DPW/license.html

Cong et al. A Latent Diffusion Model for Protein Structure Generation, 2023.

Eric Hedlin et al. Unsupervised key points from Pretrained Diffusion Models, 2024.

Introduction to diffusion models for machine learning. SuperAnnotate. (n.d.-a). https://www.superannotate.com/blog/diffusion-models

Ratford et al., Learning Transferable Visual Models From Natural Language Supervision, 2021.

Redmon et al. You Only Look Once: Unified, Real-Time Object Detection, 2016.

Rinon Gal et al., An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion, 2022.

Vaclav Kosar. Cross-attention in Transformer architecture. Vaclav Kosar's face photo. https://vaclavkosar.com/ml/cross-attention-in-transformer-architecture December 28, 2021.

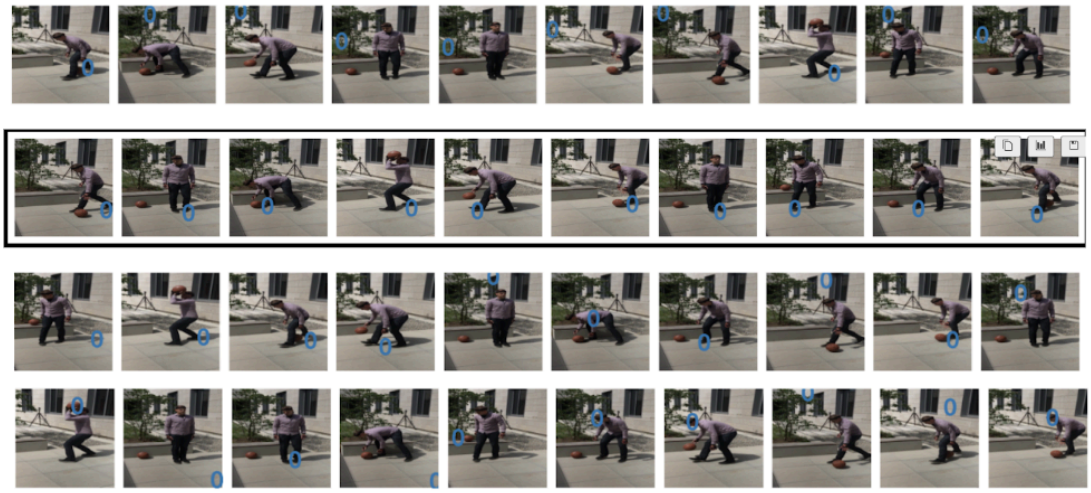
Sang Michael Xie and Sewon Min. How does in-context learning work? A framework for understanding the differences from traditional supervised learning. SAIL Blog. https://ai.stanford.edu/blog/understanding-incontext/ August 1, 2022.

Acknowledgments

Thank you to Professor Perona for giving me this opportunity and Rogério Guimarães and Markus Marks for their generous guidance, supervision, and help! I would also like to thank the Caltech VURP program and the Princeton Social Impact Internship for funding and coordinating my internship.

Appendices

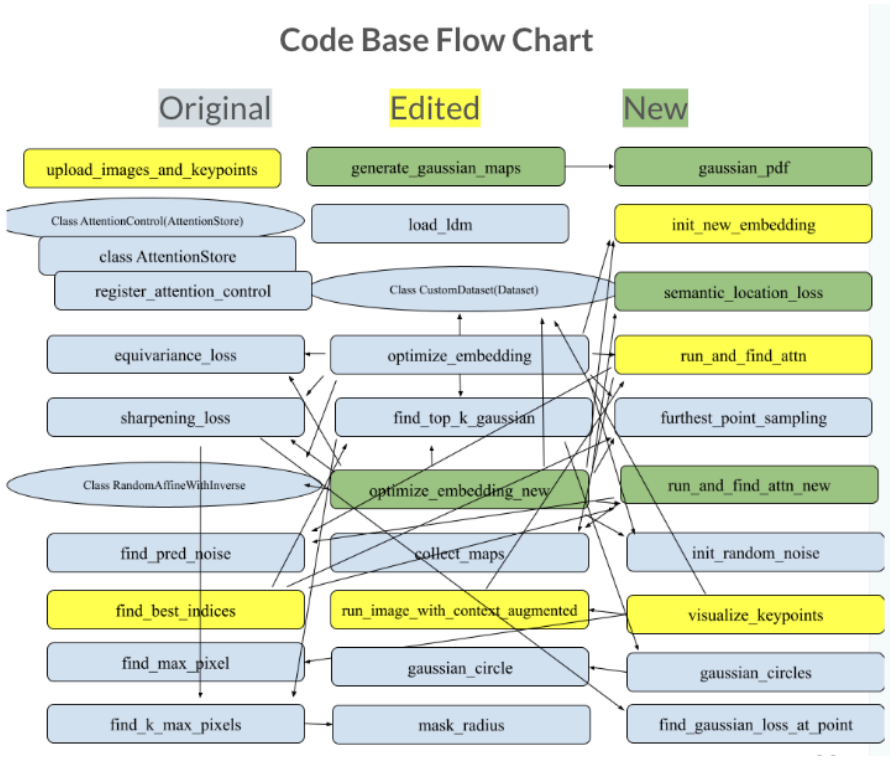
Appendix A: Examples of Left Foot Predicted Key Points with 20 Training Images



Appendix B: Examples of Head Predicted Key Points with 20 Training Images

Appendix C: Examples of Left Elbow Key Point with 20 Training Images

Appendix D: Flow Chart of Code Base



"Original" refers to the code contained in the original unsupervised key point generation paper code base (Hedlin et al.).
"Edited" refers to methods I edited.
"New" refers to methods I created for my project specifically.
The arrows refer to one function calling another function (calling the function that the arrow is pointing at).

Appendix E: What I Learned / Challenges

New skills I learned include how to understand and manipulate large and complex datasets, the theory behind latent diffusion and other computer vision techniques, how to independently problem-solve, how to communicate with others in a technical setting, how labs function, how to ask for help, and how to learn from others!

Some challenges I faced included learning the structure of the unsupervised key point generation code base (Hedlin et al.) and how to properly add to it, how to manipulate tensors, how to use a DataLoader and BatchSampler, generating ground truth attention maps that aren't in the pixel space, and general debugging.