Vision Transformers In Human Pose Estimation

Javier Salazar Cavazos, Nithin Donepudi, Wenrui Lu, Bernardo Bianco Prado
University of Michigan, Ann Arbor, Michigan, US
javiersc@umich.edu, nithind@umich.edu, wenruilu@umich.edu, beprado@umich.edu

Abstract

Human pose estimation involves visually representing a person as a graphical skeleton by representing each joint as a landmark point. This is an important task since it can be used to estimate human activity, motion transfer, and robotic movement for example. In this paper, human pose estimation is considered in a neural network model using vision transformers as implemented in the ViTPose paper [11]. For baseline comparisons, a convolutional pose machines [9] network is trained to compare vision transformers against convolutional models. More information about the code aspect can be found in Appendix C.

1. Introduction

Our goal is to solve the human pose estimation problem specifically for the upper body of a person. This problem is motivated by automotive applications in which we want to monitor whether a person driving a car is paying attention, is distracted, or is falling asleep. In the end, our goal is to use the trained model to estimate, in real time, the pose of a person on a webcam to determine driver status. However, for this project, we will only consider the first part that involves estimating pose. Determining driver status with these features is left as future work. This project is motivated by Nithin Donepudi's current independent study with a professor in the robotics department. Nithin is tasked with driver awareness detection, and this EECS504 project is a stepping stone to achieve this goal. This is the motivation for our work.

Ever since 2012, CNNs have been the de-facto model for computer vision tasks. However, over the past five years, transformers have already taken over the field of NLP, replacing LSTMs and RNNs in the process which were previously established as state of the art approaches in sequence modeling and transduction problems. When we study the architecture of vision transformers and CNNs, we find several differences in the way they process data. Vision transformers capture details/features of the entire image from the very beginning, compared to CNNs that are known to focus on small portions of the image, initially reading colors and edges working their way up to more high level features.

2. Related Work

Traditional methods for human pose estimation like pictorial structures represent the body as an undirected graph model. Specifically, it models the body as a conditional random field and analyzes the pairwise potentials between the joints [7]. This type of model was popular before neural networks took off for various computer vision tasks.

In recent years, deep learning methods relying on Convolutional Neural Networks (CNNs) achieved superior results [1]. For example, early CNN pose estimations used a ResNet backbone model in order to avoid the problem of vanishing gradients experienced with other CNN models [10]. Later models, like the popular Convolutional Pose Machine (CPM) took an approach that refined joint detection via a set of stages in the network to iteratively improve estimation by using early image features [9]. These kind of state of the art networks at the time achieved around a $\sim$60mAP score. Pose estimation then experienced rapid development in recent years by considering vision transformers in parts of the architecture. Early works tended to treat the transformer model only as a better decoder and not used as the backbone itself [6]. For example, TransPose directly processes the features extracted by CNNs as tokens in the transformer-based decoder to model the global relationship among the features [12]. These early transformer-based pose estimation methods obtain superior performance on popular keypoint estimation benchmarks. However, they either need CNNs for feature extraction which is not ideal since the receptive field is limited or requires careful design considerations of the transformer architecture due to the difficulty in training such models from scratch.

Recently, self-supervised learning methods [4] have been proposed for training plain vision transformers. With masked image modeling (MIM) as pretext tasks, these methods provide good initializations for plain vision transformers that make it easier to train. In this paper, however, we consider a vision transformer backbone model called ViTPose [11] and train from scratch to show it still outperforms CNN models such as convolutional pose machines.

3. Methods

3.1. Convolutional Pose Machine (CPM)

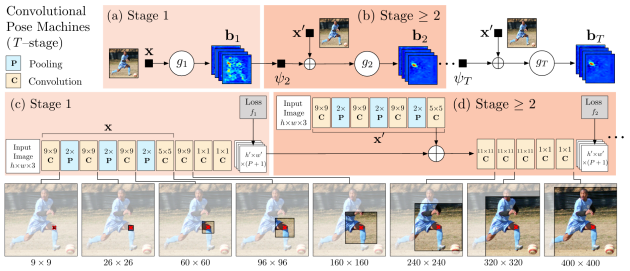


Figure 1: CPM model overview

From Figure 1 above shown in "Convolution Pose Machines" by Wei et al [9], the CPM framwork consists of convolutional layers combined with max pooling layers in a sequential manner. One set of these layers is known as a "stage". Multiple stages are linked together where early image features are added back to the output of each later stage in a manner similar to ResNet networks. The output of each stage consists of belief heatmaps for each landmark. The purpose of this is to implicitly model long-range dependencies between variables to produce increasingly refined estimates for landmark locations as the number of stages increase. Figure 2 below illustrates how further refinement in each stage increases accuracy for the estimation of the right hand landmark.

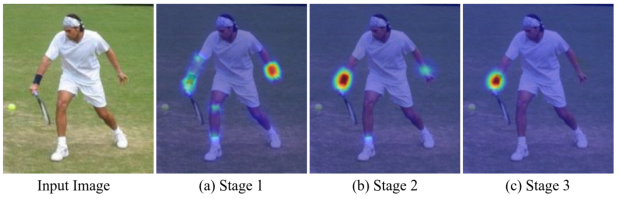


Figure 2: CPM stage refinement

An interesting observation about CPM networks is how the receptive field increases as more stages are added and thus overcome receptive field issues experienced in earlier convolutional networks that cannot see large regions of the image to gain context. Moreover, given how deep the stages go, one would naturally expect vanishing gradients to be a problem. However, since early image features are addded back to the output of each stage, this skip connection setup helps overcome the problem of vanishing gradients during training. For the project, a 6-stage CPM is trained as a baseline method to compare against ViTPose (as done in the original CPM paper).

3.2. Vision Transformer Pose (ViTPose)

An attention mechanism is a method to selectively weight the input features and learn (pay attention) to certain features based on the query the user is interested in. For more information about how attention works, refer to Appendix B. The equation for attention is described as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

where d_k is the dimension of the keys and is used to prevent small gradients. These multiple projected keys, queries, and values are fed to h attention mechanisms to get the attention matrices. Once done, all are concatenated together to be projected back to the model dimension via a linear layer. Mathematically, this can be described as the following:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$
$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

where the projection matrices are described as $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}, W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$. Here, the input data is given a positional embedding to keep track of connections of text in a sentence for example and multi-head attention is applied after layer normalization. Using ideas from ResNet [5], skip connections are introduced and can be considered a form of inductive bias. Once attention is calculated, this information is fed to a multi-layer perceptron. The transformer block is shown in Figure 13 (in Appendix B). However, these ideas have been used in language processing tasks traditionally and not in computer vision.

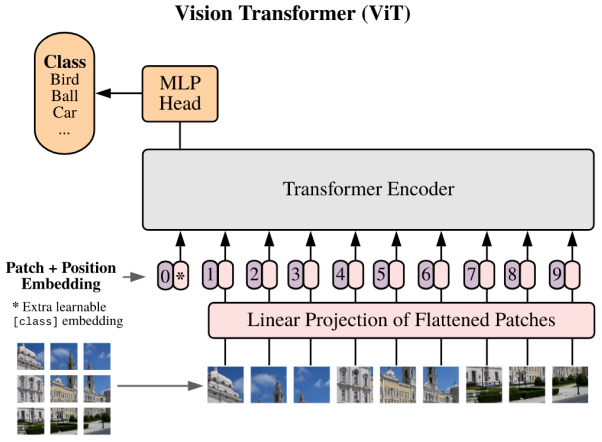


Figure 3: Vision transformer

In a influential paper by Dosovitskiy et al. [3] called "An Image Is Worth 16x16 Words", the idea of a vision transformer is introduced to adapt transformers for vision tasks.

The novelty being that patches of the images are treated as units in the same way that a word in a sentence is a unit. Breaking an image into 16x16 patches, each patch is then linearly projected to a latent dimension space concatenated with a 1D positional embedding that is learned to give positional information of the patch within the image. After going through L transformer layers, a detector head is given the final attention maps to perform various tasks like image classification. See Figure 3 above for the overall layout of a vision transformer.

Image $x \in \mathbb{R}^{H\times W\times C}$ is reshaped to patches $x_p \in \mathbb{R}^{N\times(P^2C)}$ where (H, W) is the original resolution, C is the color channels, (P, P) is the patch resolution, and $N = HW/P^2$ is the number of patches. Let $E \in \mathbb{R}^{(P^2C)\times D}$ be the linear projection matrix and $E_{\text{pos}} \in \mathbb{R}^{(N+1)\times D}$ is the positional embedding that will be learned. In a image classification problem, the vision transformer will appear as follows for $l \in \{1, ..., L\}$:

$$
\begin{aligned}
z_0 &= [x_{\text{class}}; x_p^1 E; x_p^2 E; ...; x_p^N E] + E_{\text{pos}} \\
z_l^{'} &= \text{MHSA}(\text{LN}(z_{l-1})) = z_{l-1} \\
z_l &= \text{MLP}(\text{LN}(z_l^{'})) + z_l^{'} \\
y &= \text{LN}(z_L^0)
\end{aligned}
$$

where MSHA is multi-head self attention, MLP is the multilayer perceptron, and LN is layer normalization. For this project, the ViTPose model [11] is utilized based on this idea of vision transformers as a backbone. The model is illustrated in Figure 4 along with a decoder head in Figure 5 that is standard across pose estimation and various other problems like object detection. Given a person instance bounding box $X \in \mathbb{R}^{H\times W\times 3}$ as input, the patch embedding transform such input to $F \in \mathbb{R}^{\frac{H}{d}\times\frac{W}{d}\times C}$ where d is the down sampling ratio. For each of these tokens, the data is processed by performing the following

$$F_{i+1}^{'} = F_i + \text{MHSA}(\text{LN}(F_i)) \quad F_{i+1} = F_{i+1}^{'} + \text{FFN}(\text{LN}(F_{i+1}^{'}))$$

where i represents the transformer layer $i \in \{1, ..., L\}$ and $F_0 = \text{PatchEmbed}(X)$ is the initial projection. The output of the transformer layers stays the same so $F_{\text{out}} \in \mathbb{R}^{\frac{H}{d}\times\frac{W}{d}\times C}$. After this, a standard decoder block will generate localization heatmaps for each landmark by performing the following

$$K = \text{Conv}_{1\times1}(\text{Deconv}(\text{Deconv}(F_{\text{out}})))$$

where $K \in \mathbb{R}^{\frac{H}{4}\times\frac{W}{4}\times N_k}$ is the heatmap for each landmark. N_k refers to the total number of keypoints to be estimated.

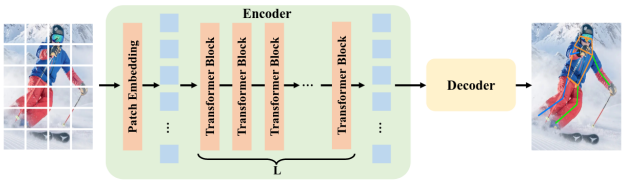


Figure 4: Model architecture

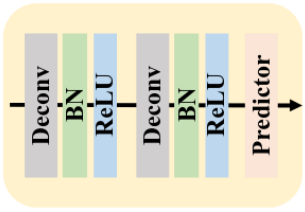


Figure 5: Decoder block

4. Experiments

For our experiments, we used the MS COCO 2017 dataset for model training. Since ViTPose is a top-down pose estimation model, we use the ground truth bounding boxes provided in the annotations. The model hyperparameters that we fine tuned in the models are provided in Table 2. These values are kept the same as the original paper values for reproducibility sake where possible. For the loss function, we use the classical loss that compares the estimated belief maps with the true belief maps generated by forming a Gaussian around each joint. This is expressed as:

$$L = \sum_{p=1}^{P} \sum_{x,y\in\mathbb{R}} \|b^p(x, y) - b_*^p(x, y)\|_F^2$$

for body part $p \in \{1, ..., P\}$ and image coordinates $(x, y) \in \mathbb{R}^2$. Figure 6 illustrates the results of training the full 12 layer model from the paper. More figures are shown in Appendix A for the other models.

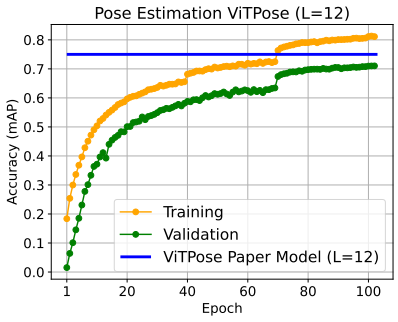


Figure 6: Accuracy plot for our ViTPose ($L = 12$) model

As a side note, for training, the data loader pipeline used involves the following: shifting the ground truth bounding boxes, randomly flipping some images, keeping only the top half of the body or lower body for some images, scaling and rotating images, normalizing tensors, and generating the target heatmaps given the point values in the annotation files. We compared our results with the ViTPose paper and CPM paper as seen in Table 1. Surprisingly, we see that the scaled-down version of ViTPose (6 layers, 43M params.) performs similarly to our 12 layer implementation (86M params.) of the model. Thus, it illustrates the advantage of smaller transformer models on pose estimation which is not discussed in the original paper. Further, we see that our implementation (12 layers) under performs the ViTPose paper model (12 layers). The most evident reasons are that we ran our model for half the total epochs and the authors used a custom weighted ADAM optimizer with layer decay that likely improved results. Moreover, one possible reason for the under performance of our model is that, due to memory limitations and batch size tradeoffs, we used FP16 optimization in our model as opposed to the standard FP32 from the paper, which could also have affected our results. Regardless, both ViTPose implementations outperform the baseline method (6-stage CPM) we trained which achieved a 59mAP score.

One can see that our implementations perform well when using images from diverse scenarios such as as person driving a car and performing physical activities as seen in figure 7. From the images, it is clear all models handle occlusions quite well, for example, when the person is facing sideways. Since these models are top-down models, it is necessary to perform object detection beforehand. We utilize the YOLOV3 objected detection model that is publicly available. This model has been trained already and we utilize it to generate bounding boxes for each human subject in the image so other objects are not considered since MS COCO is purely a human dataset for this task.

5. Conclusion

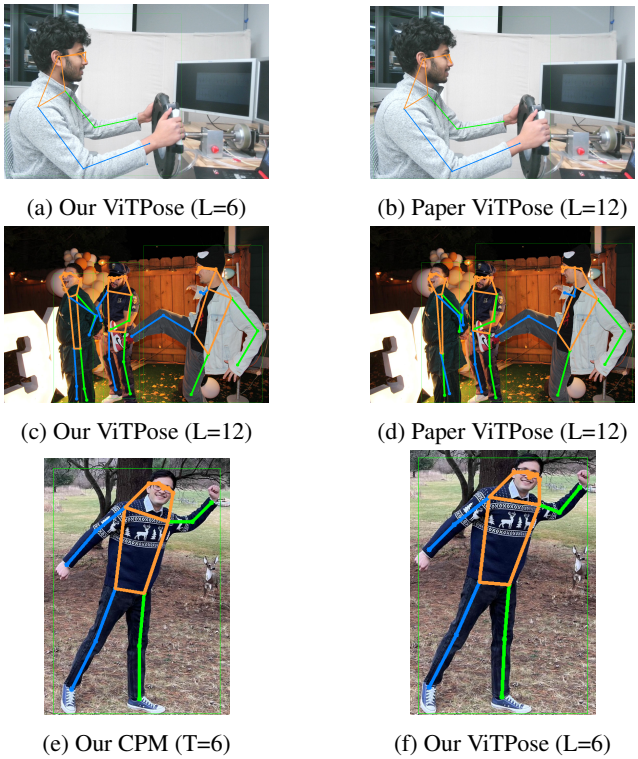
From our experiments, we observed the advantages of transformer models versus convolutional models for pose estimation. Interestingly, even a small transformer model (6 layers) can perform competitively to a full 12 layer model. We observed that our CPM results [9] were perfectly reproducible, whereas our results for ViTPose got close to the ViTPose paper results [11]. This slight discrepancy highlights the beneficial impact of using customized optimizers, as well as the fact that running a model for many more epochs can achieve a small but meaningful increase in accuracy. For future work, we seek to apply our model, along with other vision features, to determine driver distraction status in vehicles which is an important task for AI-assisted driving and crash prevention systems.

Backbone	AP	AP_{50}	AR	AR_{50}
Our ViTPose (L=6)	67.1	90.3	70.8	91.0
Our ViTPose (L=12)	71.0	91.5	74.5	92.3
Paper ViTPose (L=12)	75.8	90.7	81.1	94.6
Our CPM (T=6)	59.5	86.3	63.4	87.8
Paper CPM (T=6)	62.3	85.9	68.6	90.3

Table 1: Ablation study of the algorithms

	Backbone		
	Our VitPose (L=6)	Our VitPose (L=12)	Our CPM (T=6)
Total Epochs	120	103	80
Learning Rate	5e-4, 5e-5, 5e-6	5e-4, 5e-5, 5e-6	5e-4, 5e-5, 5e-6
Optimizer	AdamW	AdamW	Adam
Betas	(0.9,0.999)	(0.9,0.999)	-
Weight Decay	0.1	0.1	-
Image Size	(256,192)	(256,192)	(256,192)
Loss	Joints MSE	Joints MSE	Joints MSE
Heatmap Size	(48,64)	(48,64)	(24,32)
Patch Size	(16,16)	(16,16)	-
Patch Embedding	768	768	-
Heads Per Layer	12	12	-
Dropout Rate	0.0	0.0	-
Decoder Kernel Size	(4,4)	(4,4)	(4,4)
Deconvolution Layers	2	2	3
Feature Channels	-	-	128

Table 2: Training parameters for our implementations



(a) Our ViTPose (L=6) (b) Paper ViTPose (L=12)

(c) Our ViTPose (L=12) (d) Paper ViTPose (L=12)

(e) Our CPM (T=6) (f) Our ViTPose (L=6)

Figure 7: Model comparisons

References

[1] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields, 2016.

[2] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. Multi-head attention: Collaborate instead of concatenate, 2020.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[4] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[6] Ke Li, Shijie Wang, Xiang Zhang, Yifan Xu, Weijian Xu, and Zhuowen Tu. Pose recognition with cascade transformers, 2021.

[7] Leonid Pishchulin, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. Poselet conditioned pictorial structures. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 588–595, 2013.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[9] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines, 2016.

[10] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking, 2018.

[11] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose: Simple vision transformer baselines for human pose estimation, 2022.

[12] Sen Yang, Zhibin Quan, Mu Nie, and Wankou Yang. Transpose: Keypoint localization via transformer, 2020.

6. Appendix C: Project Code

Code for this project can be found in this Google Drive link: click here. This includes training files, inference files, network weights, and everything else including the README file. The backbone of both networks (CPM & ViTPose) was written by us and other things like the COCO data loader was taken from the mmpose toolbox located here. Moreover, we use mmpose to make training easier by specifying optimizer, training scheme, total epochs, etc... More details can be found in the project README file.

7. Appendix A: Additional Results

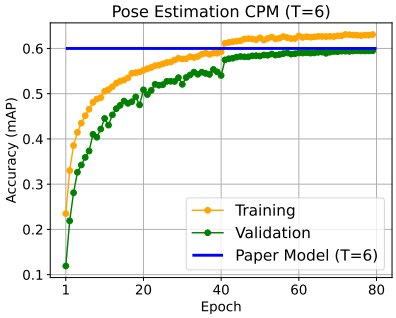


Figure 8: Accuracy plot for our CPM ($T = 6$) model

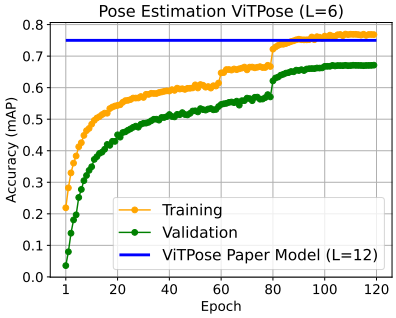


Figure 9: Accuracy plot for our ViTPose ($L = 6$) model

8. Appendix B: Attention Mechanisms

In "Attention Is All You Need" paper by Vaswani et al. [8], the transformer architecture is introduced for natural language processing based strictly on attention mechanisms with no convolutions or recurrences. An attention mechanism is a method to selectively weight the input features and learn (pay attention) to certain features based on the query the user is interested in. Visually, figure 10 from the original paper shows the attention mechanism that is broken down as inputs Q,K,V meaning the query, keys, and values.

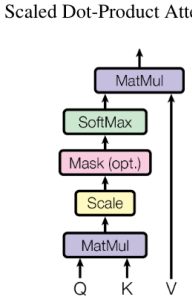


Figure 10: Attention mechanism

These queries, keys, and values are directly related to concepts from retrieval systems. Using Youtube as an example, the engine will map your query and compare a set of keys like video title or description text associated with a set of videos stored on the site, and then present the best matching videos which are the values. Mathematically, the queries and keys are multiplied to get the dot product between a key and query. If they are similar, then the value will be high given a vector interpretation. A SoftMax operation is applied to return what is essentially the most important connections. This is used to weigh the values so a small output for a certain value means it is not important for that query. The equation is described as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

where d_k is the dimension of the keys and is used to prevent small gradients. This attention mechanism is a single linear view of the sequence. In practice, multi-head attention is used where each head is an independent linear view of the sequence. Since they are independent, the hope is that each head will learn something different about the data and thus useful to improve results. Figure 11 illustrates how multiple heads are added to the attention mechanism.

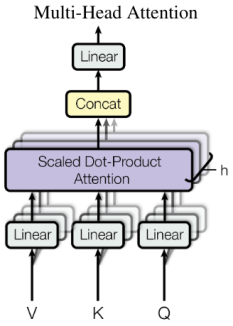


Figure 11: Multi-head attention mechanism

Each head will find a linear projection to a smaller space from the model dimension. These multiple projected keys, queries, and values are fed to h attention mechanisms to get the attention matrices. Once done, all are concatenated together to be projected back to the model dimension via a linear layer. Mathematically, this can be described as the following:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$
$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

where the projection matrices are described as $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}, W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$. Using these multi-heads is useful since figure 12 shown in Cordonnier et al. [2] illustrates that a multiple head setup will learn a low dimensional subspace of the data itself.

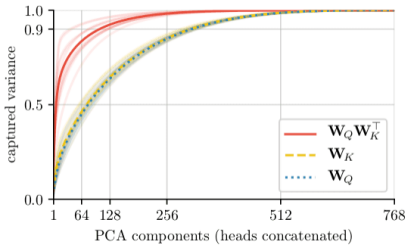


Figure 12: Captured variance of heads concatenated

Even though the output of each attention is itself not low rank, it turns out that the product after concatenation is low rank meaning the heads share common projections and focus on the important relationships of the data. This multi-head attention mechanism is a part of what is known as the transformer layer. Once attention is calculated, this information is fed to a multi-layer perceptron. Figure 13 shows the layout of a transformer block in completion. Figure 14 shows the attention output after many transformer layers in the problem of image classification.

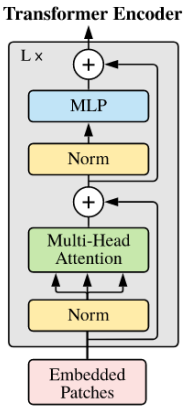


Figure 13: Transformer block

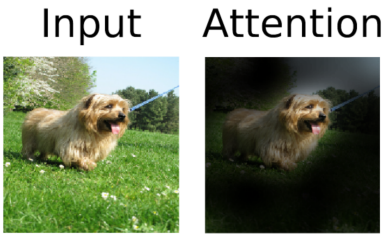


Figure 14: Attention maps