Visual Attention to Intelligent Agents

Shubham Chitnis
190020033

Vishal Srivastava
19D110025

Vishesh Agarwal
190020132

***Abstract*—Here we try to replace the existing Convolutional architecture in a Deep Q-learning agent with the current widely researched architecture i.e. Transformers. Both the models are trained to play 'Pong' a famous Atari game. Subsequent results are documented and remarks have been made on the same.**

***Index Terms*—convolution, attention, atari, reinforcement**

I. Introduction

Recently a lot of buzz has been created in the research community around transformers [1]. It is safe to say that they have successfully replaced the previous state of the art LSTM/RNN models in Natural language processing. Models such as the GPT series, BERT, AlBERT and many more have proven that inclusion of attention blocks and position embeddings are enough to overthrow older models especially in large scale tasks.

One reasoning for why such a less intuitive model works is because of how general its architecture is. Whenever we use our intuition to make architectures such as LSTMs which take inspiration from the human reading style, we introduce an inductive bias. This doesn't exist in Transformers due to which they are highly scalable and continue to perform well as long as sufficient data is fed to them.

After language, researchers started implementing them in vision which gave rise to the Vision Transformers(ViT) [2] in Oct 2020. The model attained results comparable to its CNN counterparts and required significantly less computations. A lot of research is being carried out in this field with the recently published DINO [3] giving state of the art result in self supervised segmentation tasks and 80.1% top-1 accuracy on ImageNet.

Thus we wanted to implement the Vision models in a Reinforcement Learning context. Final code is made available here. We have referenced the sources from where we have taken parts of our code [4] [5]. It should be noted that model architecture for ViT and DQAgent were already available, we had to tailor their code to make those two compatible.

II. Related Work

Very little work has been done which involves using transformer architecture to play Atari games. Some researchers from CMU and DeepMind submitted a paper to ICLR 2020 [6] which starts off with the hypothesis that transformers are unstable to train and propose methods to further stabilize it owing to which they surpass LSTMs and achieve state of the art. Another paper from DeepAI suggests the use of transformers to play Cartpole, an Atari game [7]. However none of them involve using vision models and rely on the language counterparts.

Fig. 1. Example frames from Pong

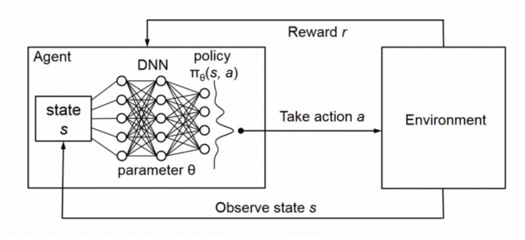

Fig. 2. Functioning of DQN

III. Approach

A. Environment

We use the python library Gym to generate our environment for Pong. We have a 84x84 frame which has both the paddles and a ball which goes back and forth. One of the paddle is hard coded and our intelligent agent plays against it. An episode denotes an entire game of pong. At a time we consider a fixed number of frames of an episode to determine optimal action. Actions here being going left, right, forward and back. Refer Fig.1 for an example setup of Atari Pong.

B. DQN

A Q-learning algorithm is used which makes use of cleverly chosen loss functions and reward estimation to come up with an optimal policy to maximize long term reward [8]. The algorithm is not fully greedy which prevents us to get stuck with a sub-optimal policy. This agent is trained on both CNN and ViT architectures and average reward us calculated per episode as a performance metric. A rough schematic has been given in Fig.2 which provides sufficient insight on how Q-learning works.

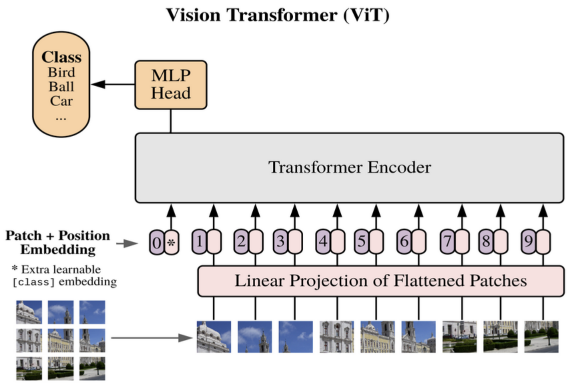
Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches

Fig. 3. ViT architecture

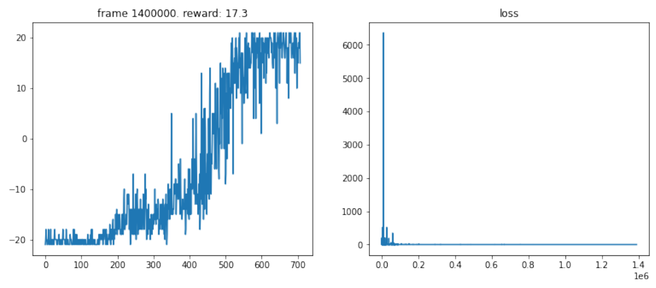

Fig. 4. CNN loss and average reward

C. ViT

Frames from an episode are further divided into smaller patches which pass through a position encoding layer. After converting each patch into a vector embedding we pass them through blocks of multi-head attention layer followed by MLP layer. Inputs undergo layer normalization and activation function used here is GeLU. Finally the activations pass through an MLP head to give us the output. This model being more general than a traditional ConvNet performs better over scale.

D. Training

A Google Colab environment is used to train three different models. One being the original CNN model and two ViT models with different hyperparameters. A batch size of 32 was used to train 1.4 million episodes. We used an Adam optimizer with 10^{-5} learning rate. All training was done on a Tesla P4 GPU provided by Colab which provided us with an average speed of 1.5-2 episodes per second in the case of ViT amd more than that for the CNN model.

IV. Results and Discussion

Performing multiple experiments wasn't possible for us due to the computational restrictions. Plots of average reward and loss have been provided for the CNN model and both ViT architectures. It can be clearly seen that after a brief period of trials, the CNN model starts improving with the reward finally reaching 14.8 after training for 1.4 million episodes. On the other hand, for both ViT models, even after training for a sufficient time we don't see any significant improvement in the reward. Rare instances of improvements are causes of randomness as we can see the average dropping in later episodes.

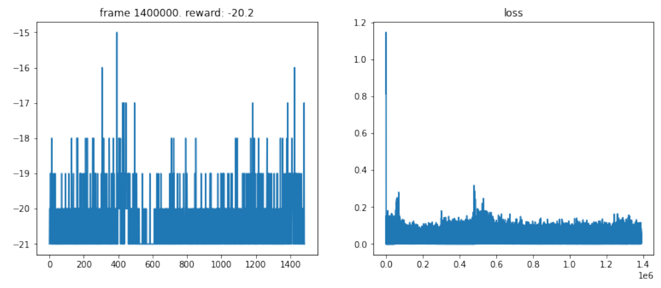

Fig. 5. ViT model 1 loss and average reward

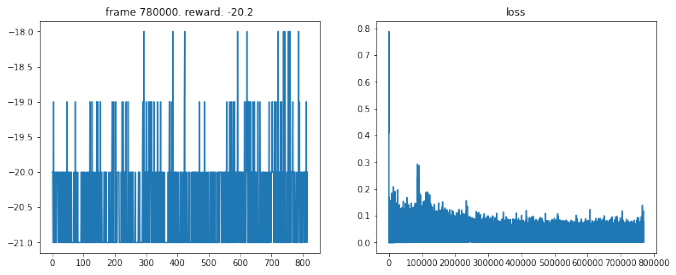

Fig. 6. ViT model 2 loss and average reward

Sudden jumps in loss for the CNN model can be accounted to the cases where model tried to explore a new path but ended up performing worse but such cases aren't visible for ViTs. We think that the reason why ViT models to underperformed is because they heavily rely on learning a good embedding layer to convey the right features from an image patch. In the case of image classification, our dataset includes images from various classes which makes it easier to learn a rich embedding. However this isn't the case with pong due to which further layers receive a bad representation of our original image. This doesn't happen in CNNs as we are directly operating on the parent image. It might be the case the with a particular choice of hyperparameters we might get a good policy, but we didn't have sufficient time and computational power to work on those.

Another interesting idea to try would be pre-training our ViT model, especially the position embedding layer on a different task such as image classification and further fine-tuning it on Pong episodes to check how much influence good embeddings have on model performance.

V. Conclusions and Future Work

From the above results we conclude that ViT models fail to come up with a good strategy as compared to their CNN counterparts in the game Atari Pong. According to us, the main cause for the same being poor embeddings and complex architecture. We have provided possible directions to look into to achieve better results. Also we only have results for Pong but similar such analysis can also be done on other Atari games too which might lead to interesting results.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.

[3] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” *arXiv preprint arXiv:2104.14294*, 2021.

[4] P. Wang, “vit-pytorch,” https://github.com/lucidrains/vit-pytorch, 2020.

[5] D. Yerzat, “Rl-adventure,” https://github.com/higgsfield/RL-Adventure, 2018.

[6] E. Parisotto, F. Song, J. Rae, R. Pascanu, C. Gulcehre, S. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury *et al.*, “Stabilizing transformers for reinforcement learning,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 7487–7498.

[7] U. Upadhyay, N. Shah, S. Ravikanti, and M. Medhe, “Transformer based reinforcement learning for games,” *arXiv preprint arXiv:1912.03918*, 2019.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.