
EFFICIENT, EFFECTIVE, EXTREMELY CHALLENGING: DEEP LEARNING NEURAL NETWORKS WITHIN A LIMIT




ABSTRACT

This paper proposes using a partially convolutional neural network to classify images within the AddNIST dataset and using a deep diffusion model utilising DDPM and UNet to generate images based on the CIFAR100 dataset. Both of these approaches have been limited within strict parameters, meaning "cheaper tactics", such as training for more steps or increasing the parameter count are no longer viable. In order to achieve passable results, more creative methods were required in order to maximise results, putting great emphasis on the architecture of the neural network as opposed to just increasing the training steps or parameter count.

Part 1: Classification

1 METHODOLOGY

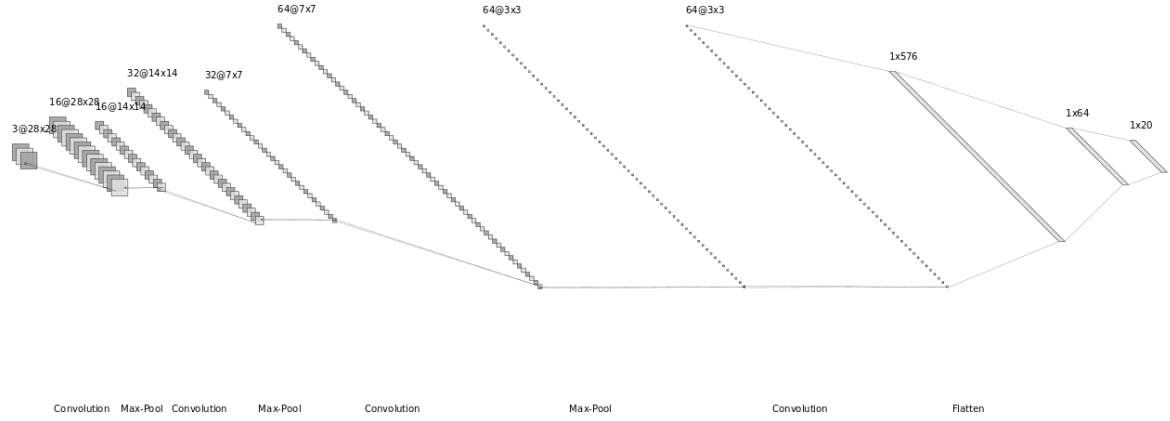
The methodology for this classifier is relatively straightforward, using a modified version of Carlos Santiago Bañón's MNIST Digits classifier, which can be found on his GitHub, [here](#). 

By using 4 convolutional layers, the features get extracted from the model without the need for any fully connected layers. In order to then properly classify these features at the end, there are 2 linear layers to reduce the dimensionality from 576 down to 64, then subsequently down to 20 to represent each class.

Between each convolutional layer is a $2 * 2$ max pooling algorithm to pick out the most prominent features from each layer. Each convolutional layer also features ReLU activation, which is scale-invariant and efficient as it only features addition and comparison operations. [1] This allows the algorithm to function relatively quickly, as it can train for 10,000 steps in around 10 to 20 minutes.

Another step taken to improve accuracy was to disable the use of TensorFlow32 units, as NVIDIA uses these to more quickly estimate tensors using a floating point estimation for each tensor. This resulted in faster computations but less accuracy. In order to maximise the accuracy, given the restrictions, the usage of TF32 had to be disabled.

Below is the architecture diagram.

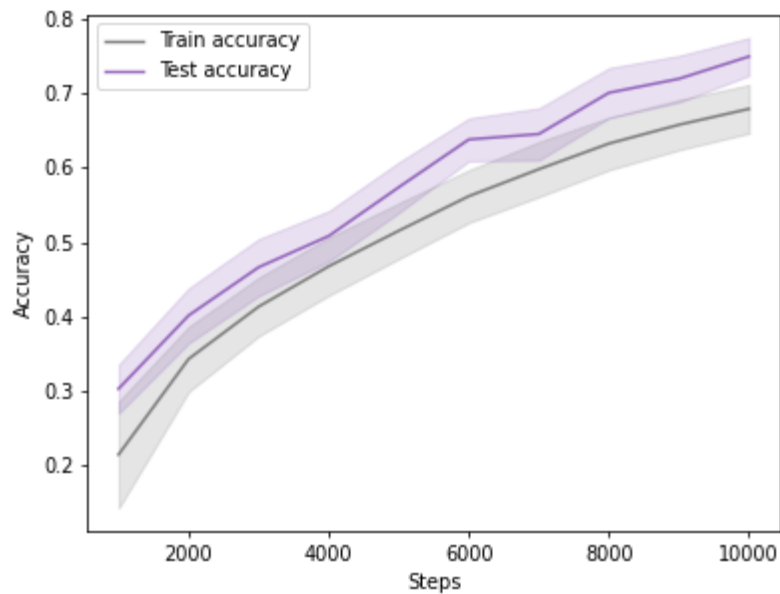


2 RESULTS

The network has 98,740 parameters, all of which are trainable.

After 10,000 optimisation steps, the model gets 67.9% training accuracy, and a 74.9% testing accuracy, illustrating its resilience to unseen data. Training loss: 1.478, train acc: 0.679 ± 0.033 , test acc: 0.749 ± 0.025 .

Here is the training graph:



3 LIMITATIONS

While the results are good for a limited network, there are many other networks with similar parameter counts but with much higher accuracy. While "porting over" Bañón's convolutional network from their TensorFlow implementation to PyTorch, I realised PyTorch could be limiting if this model were to scale as PyTorch lacks the graph optimisations that TensorFlow features since it uses dynamic graphs as opposed to the static graphs of TensorFlow. This results in more memory usage and a slightly less efficient running of the model.

In addition to this, networks with smaller convolutional dimensions but more layers may perform better than this model, as this may allow the model to pick up on more subtle features. However, this is also likely to increase the parameter count, which was not worth exploiting too much within the parameter limits given for the assignment.

Part 2: Generative model

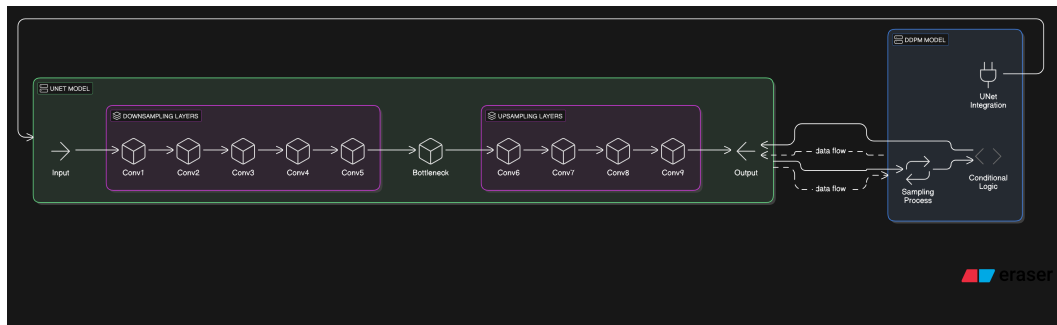
4 METHODOLOGY

The method is to train a diffusion model, using both DDPM and UNet together to condition the model based on the 100 class labels found in the CIFAR100 dataset. This DDPM algorithm is based on the algorithm seen in a 2020 paper by Ho et al. [2], in which they state that "the forward process of diffusion models is fixed to a Markov chain that gradually adds Gaussian noise according to a variance schedule $\beta_1, \beta_2, \dots, \beta_T$."

From this noise addition, the idea is for the model to "learn" what noise patterns arise within particular images, allowing it to run a backward pass to get from Gaussian noise into a legible image.

Alongside the optimiser, the model also uses a PyTorch Advanced Mixed Precision (AMP) GradScaler, which prevents underflow by multiplying network losses by a scaled factor and invoking a backward pass on the scaled loss. This means that smaller gradients do not round down to zero and are therefore always considered.[3]

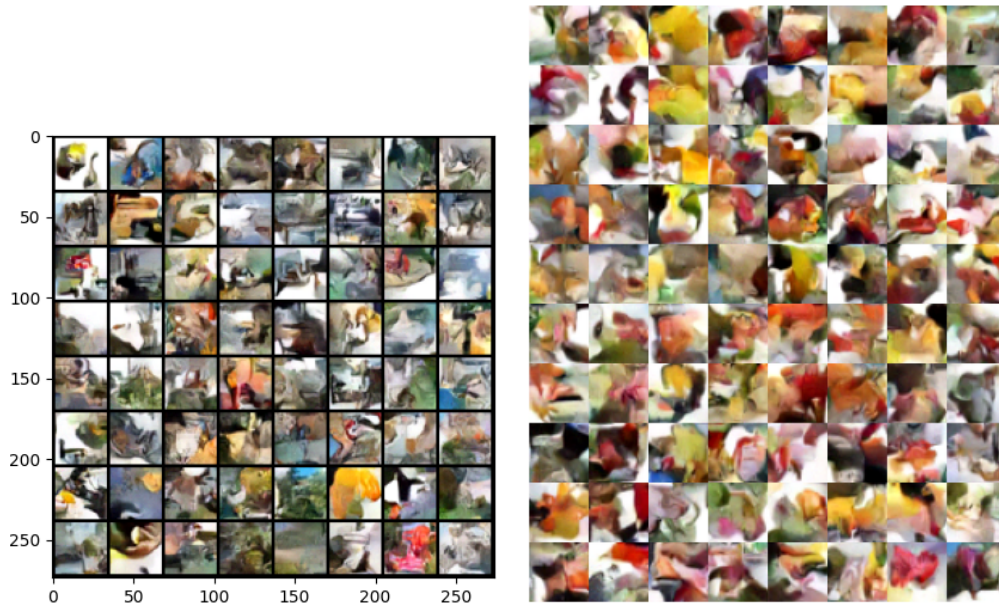
Below is the architecture diagram for this network, created using Eraser.io's graph generator, which you can find here. [🔗](#)



5 RESULTS

The network has 948,856 parameters and achieves an FID of 91.78 against the CIFAR-100 test dataset. It was trained for 50,000 optimisation steps.

The results are in a sort of "uncanny valley" stage, where the images seem like vague blobs of correct colours that vaguely resemble images seen in the training dataset. The left image shows 64 random, non-cherry picked samples, and the right image shows 10 spherical linear interpolations between 8 pairs of samples.



Most notable of these interpolants are Column 1, which is Cockroach to Fox; Column 4, which is Rose to Bowl and finally Column 6, which is Forest to Lizard. Due to the nature of a diffusion model, it becomes hard to smoothly interpolate between the images, which has resulted in these non-smooth interpolations. Other models show the difference between a sample and the "noise" from the forward pass of a sample, showing a gradient of how to get from noise to an image, but due to the nature of this model's latent interpolation sampling, this was not achievable.

6 LIMITATIONS

Successful diffusion models are widespread across the Internet, with many mainstream generative AI models, such as Stable Diffusion or DALL-E, using this technology. However, these models usually have high parameter counts and long training loops with more than 50,000 iterations. As an example, in Deepbricks' MosaicML research [4], they replicated Stable Diffusion 2, comprised of a text encoder, variational autoencoder and a UNet diffusion model, using 790 million image-caption pairs (amounting to over 100 terabytes of data), 850,000 training iterations over 128 NVIDIA A100 graphics processors [3], with the model itself having 860 million parameters in the UNet and 123 million in the text encoder [5].

The usage of other more efficient models for a small-scale diffusion problem like this one, such as a generative adversarial network (GAN), may be better, as the generator and discriminator work in tandem to improve both parts at once. This allows the GAN to produce high-quality samples much faster than a diffusion model since they need only one pass through the process. Regarding the classic generative learning trilemma covered by Xiao, Kreis and Vahdat in their 2022 paper [6], GANs cover "high quality sampling" and "fast sampling", while diffusion models cover "high quality sampling" and "mode coverage / diversity". For larger models, the instability of a GAN tends to be amplified as they can suffer from mode collapse. However, for a small task like this one, with only 100 different class labels, the reward of having high-quality images produced in fewer training iterations outweighs the potential risk of mode collapse during the training process.

REFERENCES

- [1] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík.

-
- Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 315–323. URL: <https://proceedings.mlr.press/v15/glorot11a.html>.
- [2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: <https://arxiv.org/abs/2006.11239>.
 - [3] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
 - [4] Mihir Patel et al. *Training Stable Diffusion from Scratch for under \$50k with MosaicML* — *databricks.com*. <https://www.databricks.com/blog/stable-diffusion-2>. [Accessed 05-02-2025]. 2023.
 - [5] Wikipedia contributors. *Stable Diffusion* — *Wikipedia, The Free Encyclopedia*. https://web.archive.org/web/20250205115917/https://en.wikipedia.org/wiki/Stable_Diffusion. [Online; accessed 5-February-2025, archived 5-February-2025]. 2025.
 - [6] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. *Tackling the Generative Learning Trilemma with Denoising Diffusion GANs*. 2022. arXiv: 2112.07804 [cs.LG]. URL: <https://arxiv.org/abs/2112.07804>.