PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100-ft Ring Road, Bengaluru –560 085, Karnataka, India

A Project Report
On

Implementation of Image Generative Models Using ImageGPT

Submitted in fulfillment of the requirements for the
Project phase -1

Submitted by
Payel Dutta
SRN: PES2202100950

Under the guidance of
Kaustuv Kunal
Principal Data Scientist, Great Learning

February-April 2023

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROGRAM - M. Tech in Data science and Machine Learning

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROGRAM - M. Tech in Data science and Machine Learning

CERTIFICATE

This is to certify that the Dissertation entitled

'Implementation of Image Generative models using ImageGPT'

is a bonafide work carried out by

Payel Dutta
SRN: PES2202100950

In partial fulfillment for the completion of 4th semester coursework in the Program of Study MTECH in Data science and Machine Learning under rules and regulations of PES University, Bengaluru during the period Feb 2023 – Apr. 2023. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the 4th semester academic requirements in respect of project work.

Signature with date & Seal	*Signature with date & Seal*	*Signature with date & Seal*
Internal Guide	Chairperson	Dean of Faculty

Name and Signatures of the Examiners

1.

2.

3.

DECLARATION

I hereby declare that the Project Phase - 1 entitled "**Implementation of Image Generative models using ImageGPT**" has been carried out by me under the guidance of **Kaustuv Kunal** and submitted in partial fulfilment of the course requirements for the award of degree of **Master of Technology in Data Science and Machine Learning** of **PES University, Bengaluru** during the academic semester Feb 2023 – Apr 2023. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

SRN Number **Student Name**

Abstract:

Motivated by advancements in unsupervised representation learning for natural language, this paper sought to investigate whether comparable models could also generate useful representations for images. Using a sequence Transformer, I trained the model to autonomously predict pixels, without any awareness of the 2D structure of the input. Despite being trained on low-resolution ImageNet without labels, the GPT-2 scale model achieved impressive results, displaying robust image representations, as evidenced by linear probing, fine-tuning, and classification with low-data. To achieve the desired outcome, I implemented a two-stage approach, involving pre-training and fine-tuning. During the pre-training stage, I experimented with both auto-regressive and GPT objectives. In addition the sequence Transformer architecture was used to predict pixels instead of traditional language tokens. Based on findings, it appears that generative image modeling remains a promising avenue for acquiring high-quality, unsupervised image representations. This study will come in useful whenever there is a need for image completion. Given any random half image, the model will try to predict completed version of the image in multiple forms, which can be extended later to facial recognition, object identification as well as it might find its use in healthcare domain.

TABLE OF CONTENTS:

LIST OF FIGURES:

LIST OF TABLES:

1. INTRODUCTION

In today's world, human beings are very fond of clicking pictures, otherwise known as images. This is specially attributed to the improvement and affordability of technology that every family owns a smartphone or a camera. People want to capture every moment and almost $1/10^{th}$ of the world's population is addicted to taking selfies. This results in a huge amount of data when it comes to preserving the images.

The rate of crime is also increasing nowadays, and given a good resolution of image, it is easier for the cops to nab the criminal. But, oftentimes, the images might be half visible, might be partially dark, might not be clear etc. In those case, facial recognition can prove to be a daunting task. This paper aims to explore the areas wherein ImageGPT (iGPT) can help in image completion[1], provided it has already been trained by a large number of image datasets.

These days people are very social and often gather in large happening events where chances of losing individual items are very high. Those items might be very expensive or might be of value to them and they will try to put in all efforts to get it back, but often times, the picture of the exact lost item might not be available, but they might get an object which is quite similar to it. In those cases, given the newer object as an input, the model will try to predict similar images to the lost item, that will give the searchers an idea of what the item looked like. This method could also be useful for identifying missing persons.

In this paper, I tried to leverage ImageGPT for image completion. The existing papers implemented iGPT with TensorFlow. Since TensorFlow is not supported by browsers, so I implemented the model with Pytorch and to get better accuracy of the model, I augmented the training data, to generate more samples. A common approach to evaluate the quality of representations is to fine-tune them for image classification. This involves appending a small classification head to the model, which optimizes a classification objective while adapting all weights. Pre-training can then be seen as a valuable initialization or regularizer when combined with early stopping. The dataset used in this study is MNIST dataset, but this can also be extended to other datasets in the future.

A background on the project is provided in Section 1.1. A concise problem statement is described in Section 1.2, Section 1.3 introduces the materials and methods used in this study.

1.1 Background

Unsupervised pre-training played a pivotal role in the renaissance of deep learning, as methods such as the Deep Belief Network and Denoising Autoencoder gained popularity in computer vision and speech recognition in the mid-2000s. It was thought that a model that learned the data distribution P(X) would also acquire useful features for subsequent supervised modeling of

P(Y|X). However, advancements such as improved initializations, normalization strategies, and piecewise linear activation functions eventually rendered pre-training unnecessary to obtain strong results. As research showed that deep unsupervised representations may not always provide advantages over single-layer learned features or random features, the approach fell out of favor.

Instead, pre-training made significant strides in the field of Natural Language Processing after strong results were observed for word vectors. BERT, one of the most prevalent approaches, has a training objective that resembles that of the Denoising Autoencoder originally developed for images. Images are known to be more challenging for generative modeling because they are a higher-dimensional, noisier, and more redundant modality than text. However, self-supervised methods that encourage the modeling of global structures have shown promise. Advances in training objectives, more recent architectures, and greater model capacity have enabled these methods to achieve state-of-the-art performance in low-data settings and sometimes even surpass supervised representations in transfer learning scenarios.

As generative pre-training methods for images celebrate their tenth anniversary and continue to exert a substantial impact in NLP, it is time to revisit this category of methods and compare them with recent progress in self-supervised methods. This study aims to re-evaluate generative pre-training on images and assess its competitiveness relative to other self-supervised approaches. It is demonstrated that with a flexible architecture, an efficient likelihood-based training objective, and considerable computational resources (2048 TPU cores), generative pre-training can hold its own against other self-supervised methods. The findings indicate that generative pre-training is capable of learning representations that significantly enhance the state of the art in low-resolution unsupervised representation learning scenarios.

Fig 1.1 – Pretraining of Pixels

Source: OpenAI

1.2 The Problem Statement

Nowadays, due to the immense volume of images, it is difficult to effectively store all the data, and it is very likely that some images might get deleted or the quality might be compromised. It will be very difficult for the professionals, to correctly identify person or things due to lack of relevant data. ImageGPT, launched in the year 2020, was built on top of GPT-2 and has shown good performance in both image classification as well as image completion. In this paper, I wanted to explore the areas of image completion, where a model will be able to predict complete images when provided with half-cut images, or low-quality images. The pretrained model will predict multiple similar images, and then the professional will just have to pick the image that best matches the description. This will save lot of time in manually imagining/painting pictures of criminals, objects, or places as it will be readily available to them.

1.3 Image Completion

Image completion, also known as image inpainting, is a computer vision task that involves filling in missing parts of an image. This technique can be used for a variety of applications, including photo editing, restoration, and reconstruction. Image completion algorithms use a variety of approaches to fill in the missing parts of an image. Some algorithms use neighboring pixels to estimate the missing values, while others use more complex methods like deep neural networks. One popular method for image completion is called the PatchMatch algorithm, which uses a database of known image patches to fill in missing areas.

Image completion is particularly useful for photo restoration, where old or damaged photos can be restored to their original condition by filling in missing areas. This technique can also be used in photo editing, where unwanted objects or blemishes can be removed from an image by filling in the area with surrounding pixels. Another application of image completion is in the field of computer graphics, where incomplete or missing data can be filled in to create more realistic images. For example, 3D models can be completed by filling in missing areas of texture data, or by filling in gaps in the geometry of the model.

1.3.1 Process of Image Completion

iGPT, or image Generative Pre-trained Transformer, is a type of deep learning model that is primarily designed for natural language processing tasks. While GPT is not specifically designed for image completion, it can be used for this task by adapting it to work with images. iGPT has been built on top of GPT-2 and leverages all the capabilities of GPT-2 for image completion. The process of image completion can be summarized as below:

1. The process of image completion in iGPT involves training the model on a large dataset of images and their corresponding missing parts. The missing parts of the images can be

artificially created by removing random patches or by simulating damage or degradation to the image.
2. During training, the GPT model learns to predict the missing parts of an image based on the surrounding pixels. The model does this by encoding the input image into a high-dimensional vector representation, which is then used to generate the missing pixels.
3. Once the model has been trained, it can be used to complete new images. To do this, the input image is first fed into the iGPT model to generate a vector representation. The model then uses this representation to generate the missing pixels, which are added back into the image to complete it.

1.3.2 Benefits of iGPT in Image Completion

ImageGPT, also known as iGPT, is a variation of the GPT model that is specifically designed for image completion tasks. ImageGPT has several benefits when it comes to image completion, including:

1. **High-quality results:** ImageGPT can generate high-quality completed images that are visually consistent with the input image. This is because the model has been trained on a large dataset of images, allowing it to learn patterns and structures that are commonly found in natural images.
2. **Flexibility:** ImageGPT can be used for a wide range of image completion tasks, from simple object removal to more complex tasks like image restoration or reconstruction. This flexibility makes it a useful tool for many different applications, from photo editing to computer graphics.
3. **Efficiency:** ImageGPT is computationally efficient compared to some other image completion techniques, such as deep convolutional neural networks. This makes it easier to train and deploy in real-world applications.
4. **Generalization:** ImageGPT has the ability to generalize to unseen data, which means that it can generate completed images that are consistent with the input image even if it has never seen similar images before. This is a powerful feature that can be especially useful in scenarios where there is limited training data available.

In summary, the benefits of ImageGPT in image completion include high-quality results, flexibility, efficiency, and generalization. These advantages make it a valuable tool for a wide range of applications in computer vision, photo editing, and computer graphics.

1.3.3 Challenges associated with iGPT in Image Completion

While ImageGPT has several benefits when it comes to image completion, there are also some challenges associated with the model. Some of the challenges include:

1. **Large amount of training data:** ImageGPT requires a large amount of training data to achieve good results. This can be challenging for some applications where limited data is available.
2. **Computationally expensive:** ImageGPT is computationally expensive to train and run, which can make it challenging to deploy in real-time applications.
3. **Limited spatial awareness:** ImageGPT does not have a strong spatial awareness of the input image, meaning that it may struggle with completing images that have complex spatial relationships or patterns.
4. **Limited domain-specific knowledge:** ImageGPT has been trained on a general dataset of images and may not have domain-specific knowledge that is required for some applications.
5. **Limited control over output:** ImageGPT generates completed images autonomously, which means that there is limited control over the output. This can be challenging in some applications where specific output is required.

These challenges may make it unsuitable for some applications or require additional training and fine-tuning to overcome.

1.4 Proposed Solution

iGPT is a generative model that uses a transformer architecture for image completion tasks. The model works by predicting the missing pixels in an image based on the surrounding pixels. The input image is first preprocessed by resizing it to a fixed size and normalizing the pixel values. The preprocessed image is then fed into the ImageGPT model, which uses a series of transformer layers to encode the input image into a high-dimensional vector representation. A portion of the encoded image is then masked out, representing the missing pixels that need to be completed. The model then generates the missing pixels by using an autoregressive generation process. Starting from the leftmost missing pixel, the model predicts the value of the pixel based on the surrounding pixels that have already been generated. The process is then repeated until all the missing pixels have been generated. The completed image is then obtained by adding the generated pixels back into the input image.

The key to the success of ImageGPT is the use of transformer layers. These layers allow the model to capture long-range dependencies in the input image, which is essential for generating high-quality completed images. Additionally, the autoregressive generation process ensures that the completed image is visually consistent with the input image. In summary, ImageGPT works by encoding the input image into a high-dimensional vector representation, masking out a portion of the encoded image to represent the missing pixels, using an autoregressive generation process to predict the missing pixels, and adding the generated pixels back into the input image to obtain the completed image

2. LITERATURE SURVEY

The history of ImageGPT (iGPT) can be traced back to the development of the GPT (Generative Pre-trained Transformer) model by OpenAI. The GPT model was designed to generate coherent and natural language text by training a large-scale neural network on a massive corpus of text data. The success of GPT inspired the development of the iGPT model, which was trained to generate images from textual descriptions using a similar approach. The iGPT model was introduced in 2021, and it builds on previous work on generative models for images such as the Generative Adversarial Network (GAN) and Variational Autoencoder (VAE).

iGPT is based on the transformer architecture, which is a type of neural network that has proven to be highly effective in natural language processing[2]. The model is trained on large datasets of image-caption pairs, allowing it to learn the relationship between textual descriptions and the corresponding visual content. The iGPT model has shown impressive results in generating high-quality images that match textual descriptions, and it has the potential to be used in a wide range of applications, including design, art, and gaming. As with other deep learning models, ongoing research and development will likely continue to improve and expand the capabilities of iGPT in the future. To understand about iGPT, we need to know a bit about the evolution of GPT that has been explained in section 2.1

2.1 Evolution of GPT

The evolution of iGPT (imagined Generative Pretraining Transformer) is a recent development that builds upon the success of GPT in natural language processing. iGPT is a generative model that can generate diverse and high-quality images by learning from large amounts of unlabeled image data using unsupervised learning techniques.

The development of iGPT can be traced back to the success of generative adversarial networks (GANs) and variational autoencoders (VAEs), which are two popular types of generative models used in image processing. GANs and VAEs were able to generate high-quality images, but they had limitations such as mode collapse, where the model would generate only a few similar images repeatedly. In 2018, researchers at OpenAI introduced GPT-2[3], a large-scale transformer-based language model that achieved state-of-the-art performance on a range of language tasks. Building on this success, the researchers explored the possibility of using a similar architecture to generate images.

In 2021, OpenAI released iGPT, which is a generative model trained using a variant of the GPT architecture on large amounts of unlabeled image data. The model is pre-trained on a large dataset of images using a self-supervised learning approach known as contrastive predictive coding (CPC). This approach involves training the model to predict which pixel belongs to the same object or group of objects, given the context of the image.

The pre-trained iGPT model can then be fine-tuned on a specific downstream task, such as image classification or object detection. In contrast to other generative models, iGPT can generate high-quality images across a wide range of categories, styles, and resolutions without overfitting to a small set of similar images[4].

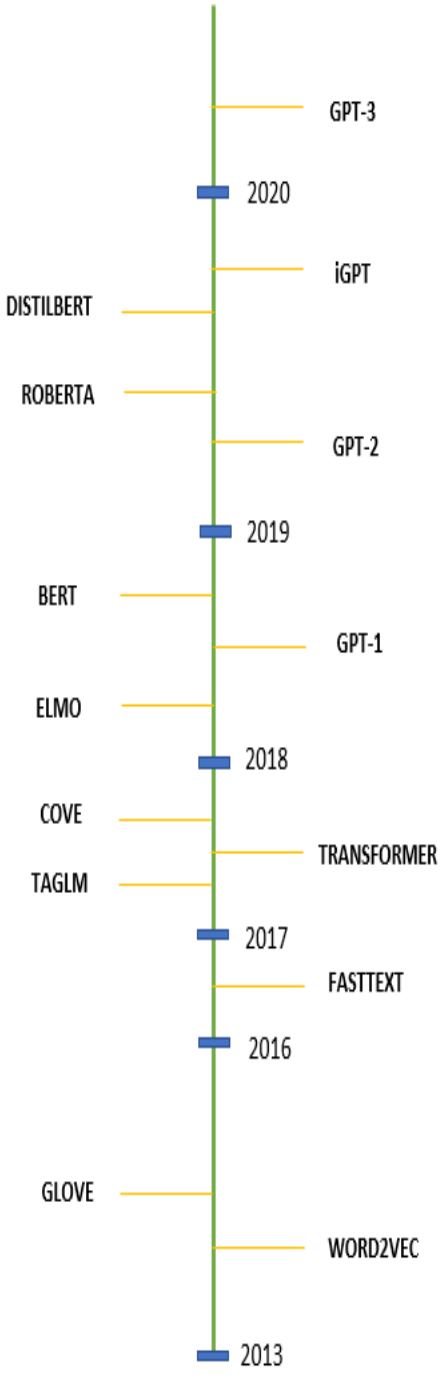

Fig 2.1: Evolution of GPT

2.1.1 History of GPT:

Established in 2015 as a non-profit organization, OpenAI initiated the development of GPT as part of its research agenda to promote and create "friendly AI" that would serve the betterment of humanity. The model was first introduced in 2018, containing 117 million parameters. The following year, OpenAI released an even more sophisticated version, GPT-2, featuring 1.5 billion parameters[5]. In comparison, GPT-3, the latest version, boasts a massive 175 billion parameters, surpassing its predecessor by more than 100 times and outperforming other similar programs by ten times.

Earlier models, including BERT, had demonstrated the potential of the text generator approach, showcasing the remarkable capabilities of neural networks in producing extensive pieces of text, which were previously deemed impossible.

To prevent any potential issues, OpenAI cautiously released access to the model in increments to observe its usage. During the beta phase, users had to apply to use the model, and access was initially free of charge. However, the beta phase ended in October 2020, and OpenAI launched a tiered credit-based pricing model, ranging from free access to 100,000 credits or three months of access to more substantial charges of hundreds of dollars per month for more significant access[6].

In the same year, Microsoft invested $1 billion in OpenAI, becoming the sole licensee of the GPT-3 model's underlying technology. This means that Microsoft has exclusive access to GPT-3's underlying model. This partnership provides Microsoft with the opportunity to integrate GPT-3 into its own products and services.

In November 2022, ChatGPT was launched, and it was initially free for public use during its research phase. This launch brought GPT-3 more mainstream attention, providing many non-technical users with the opportunity to try the technology.

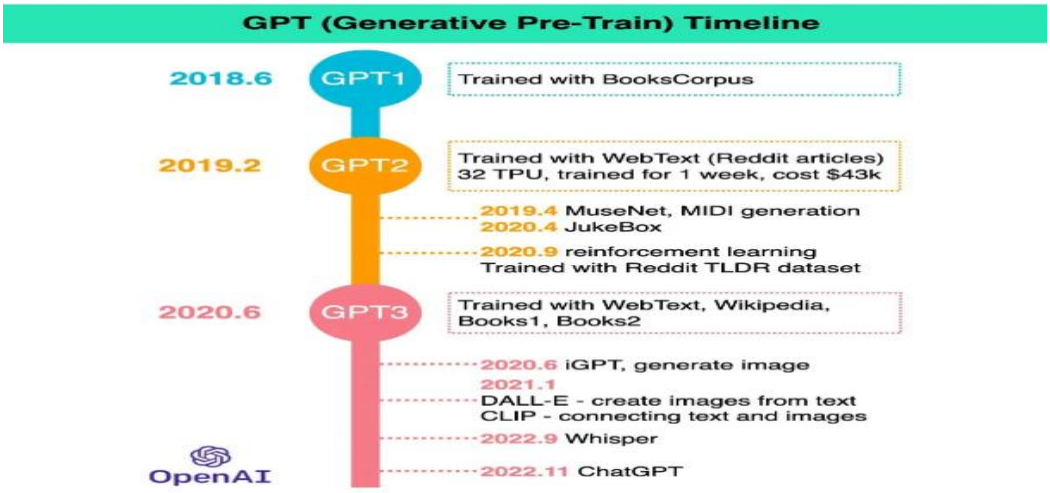

Fig 2.2: History of GPT

Source: OpenAI

2.1.2 Future of GPT:

As GPT-3 continues to gain popularity, researchers and engineers are already working on even more advanced models. Some open-source efforts aim to provide a non-licensed alternative to Microsoft's exclusive ownership of the GPT-3 model. OpenAI is also planning to release more specialized versions of their models trained on a wider variety of texts.

Different use cases and applications of the GPT-3 model are also being explored. However, Microsoft's exclusive license presents challenges for developers who want to integrate the capabilities of GPT-3 into their applications. Despite this, Microsoft has expressed interest in incorporating a version of ChatGPT into popular applications like Word, PowerPoint, and Power Apps.

It remains uncertain how GPT-3 will evolve in the future, but it is likely that it will continue to be used in various generative AI applications. Experts predict that there will be exponential technical advancements in the generative AI space, with continued investment from tech giants such as Microsoft, Google, Apple, and Nvidia[7].

GPT-4 is likely to be released in 2023, and the model will be trained on 100 trillion parameters and has the potential to surpass all GPT models that exist today.

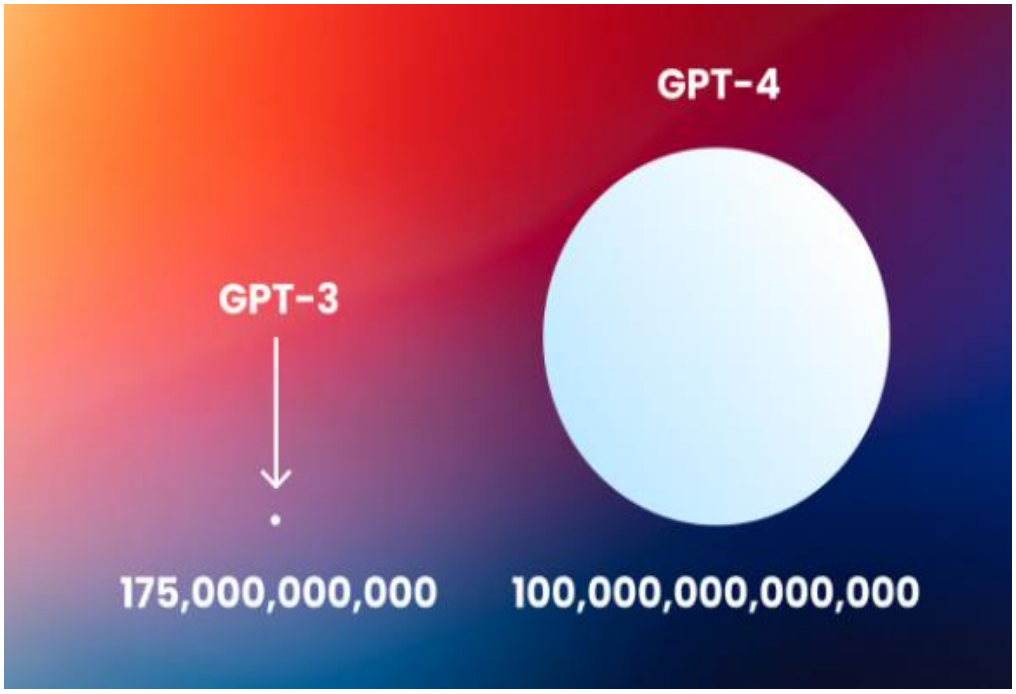

Fig 2.3: Future of GPT

Source: OpenAI

2.2 GPT models

The OpenAI team has developed a series of deep learning-based language models called GPT models. These models can perform several NLP tasks, such as question-answering, textual entailment, and text summarization, among others, without supervision. Moreover, these language models can accomplish these tasks with very few or even no examples. Surprisingly, they often perform as well as, if not better than, state-of-the-art models trained with supervision.

2.2.1 GPT - 1

OpenAI' s researchers unveiled a groundbreaking approach to natural language understanding (NLU) in 2018. Their framework involved a single model that can handle multiple tasks, achieved through a combination of generative pre-training and discriminative fine-tuning. This innovative approach had the potential to revolutionize the field of NLU and bring us one step closer to machines that can truly understand language.GPT-1 made history by becoming the first model to read text and provide answers to questions about it. Its training data consisted of Wikipedia articles, which helped GPT-1 acquire a human-like language proficiency for answering questions[8]. The advent of GPT-1 marked a significant milestone in AI development as it enabled computers to grasp textual information in a more intuitive manner than previous models. Nevertheless, despite its groundbreaking capabilities, GPT-1 has some limitations that restrict its scope of application.

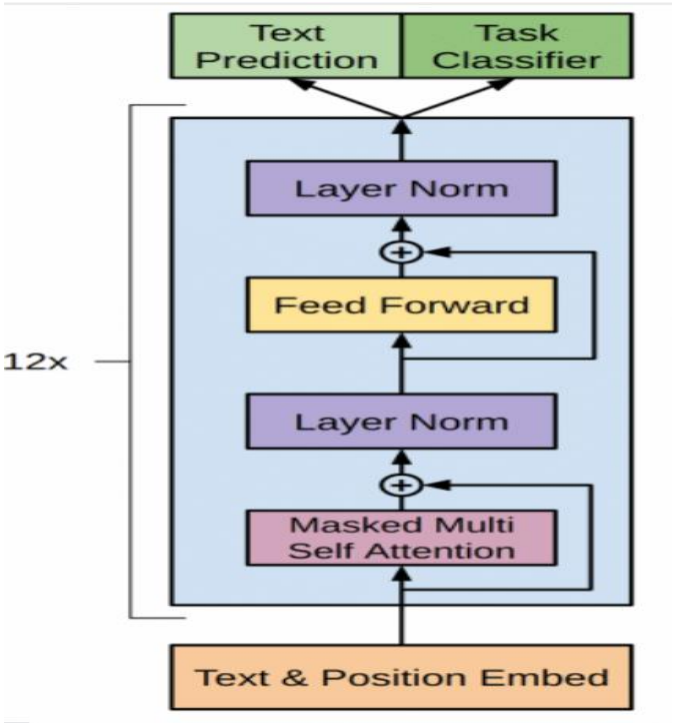

Fig 2.4 – Architecture of GPT-1 model

Source: ResearchGate

2.2.2 GPT - 2

GPT-2 is a powerful transformers model[9] that has been pre-trained on a massive corpus of English data in a self-supervised manner. Unlike supervised learning where humans annotate the data, GPT-2 learned from raw text inputs without any human intervention. This allowed the model to utilize vast amounts of publicly available data for its training. During pre-training, GPT-2 was trained to predict the next word in each sentence, which required it to learn the underlying patterns and structures of the English language.

The model operates by taking sequences of text of a specific length as inputs, and predicting the same sequence shifted by one token to the right as the target. The model uses a masking mechanism internally to ensure that it only uses the inputs from the past tokens for each prediction.

Thanks to its pre-training, GPT-2 has acquired a comprehensive understanding of the English language that can be leveraged to extract features useful for various downstream tasks. However, it excels at the task it was specifically pre-trained for, which is generating coherent and fluent texts based on a given prompt.

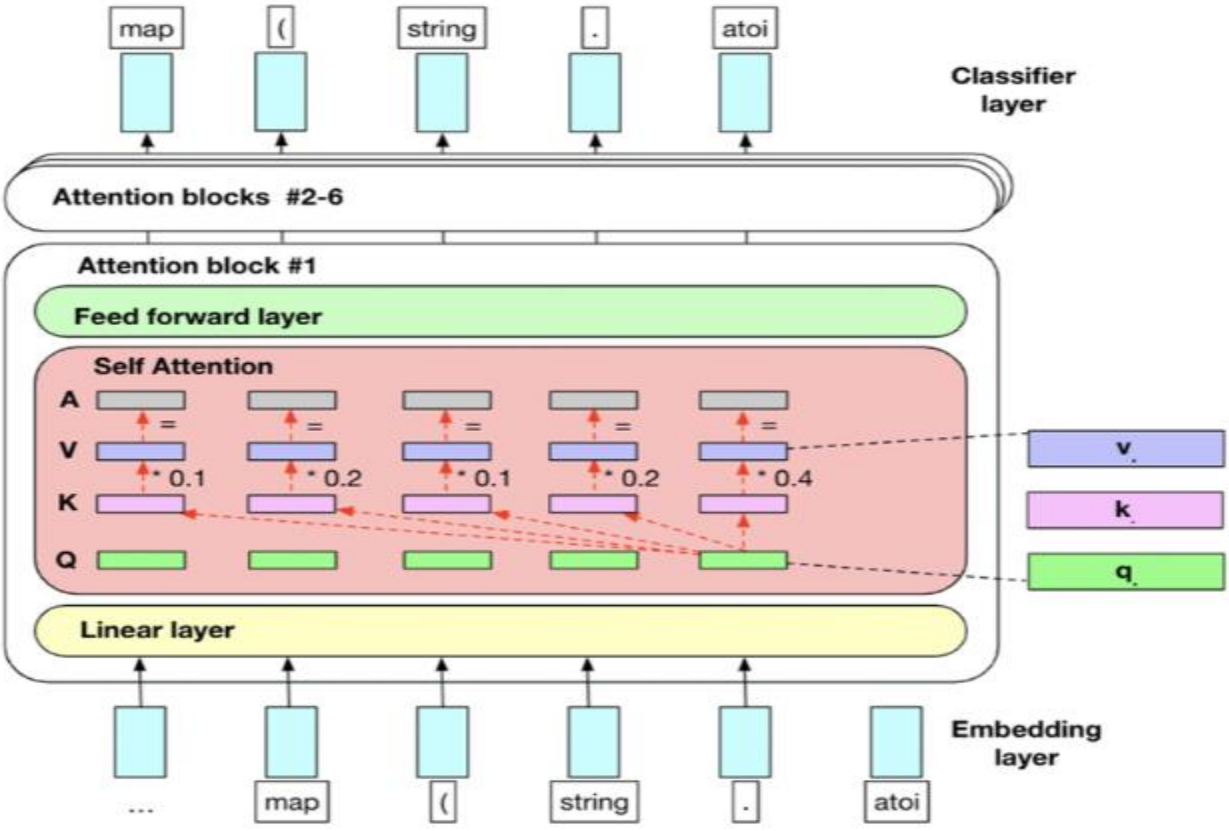

Fig 2.5 – Architecture of GPT-2 model

Source: ResearchGate

2.2.3 GPT - 3

In 2020, the world witnessed the release of Generative Pre-trained Transformer 3 (GPT-3), a remarkable autoregressive language model that uses deep learning to generate human-like text. By feeding it an initial text prompt, the model can produce a continuation of that prompt that appears as if it were written by a human.

GPT-3 has an architecture that consists of a decoder-only transformer network with an impressive context of 2048 tokens, making it one of the largest language models to date with 175 billion parameters. To achieve this size, the model requires a storage capacity of 800GB. GPT-3's training utilized generative pre-training, where it was trained to predict the next token in a sequence based on the previous tokens. This approach enabled the model to demonstrate strong zero-shot and few-shot learning on various tasks[10].

The creators of GPT-3 described how the model improved the performances of natural language processing (NLP) in language understanding by employing a process of "generative pre-training of a language model on a diverse corpus of unlabeled text, followed by discriminative fine-tuning on each specific task." By doing so, it eliminated the need for human supervision and time-consuming hand-labeling, paving the way for more efficient and effective NLP.

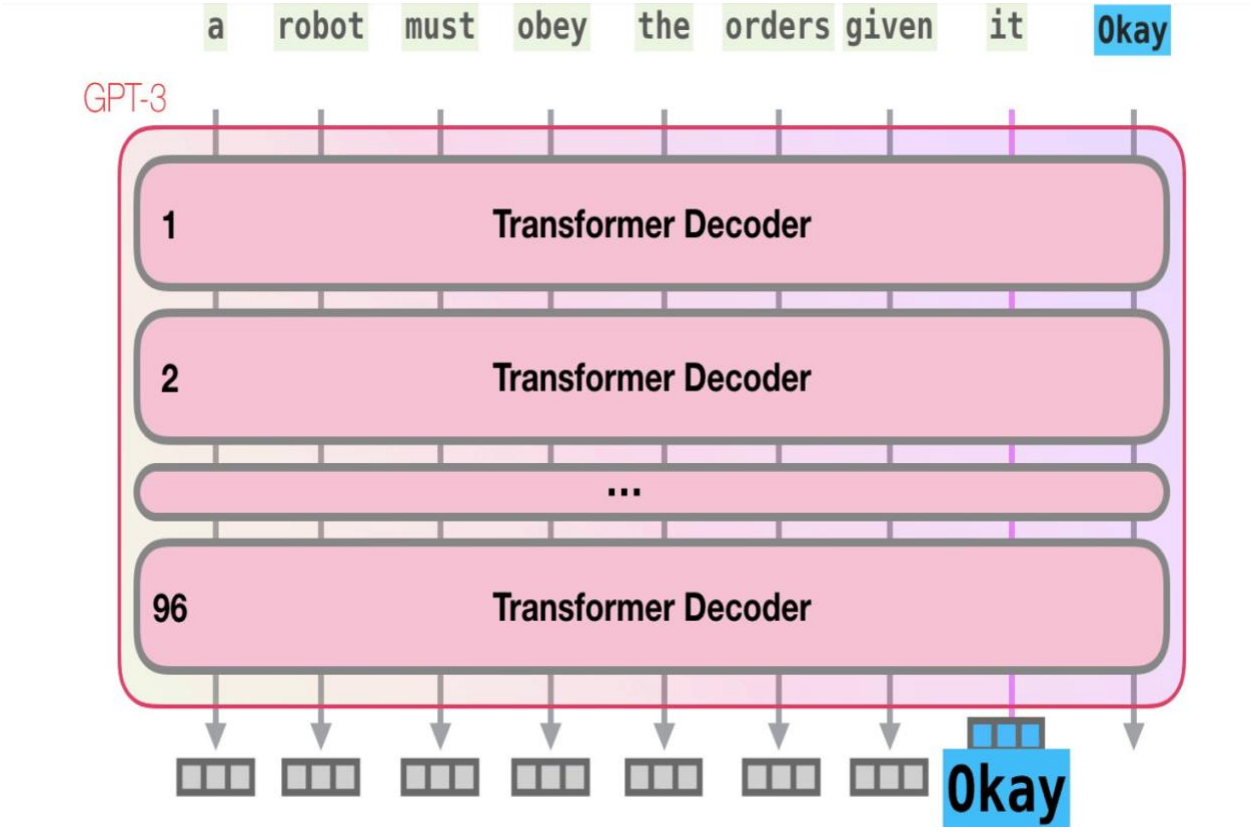

Fig 2.6 – Architecture of GPT-3 model

Source: ResearchGate

2.3 Comparison of GPT models

With the evolution of GPT, we see that each model performs better than its predecessor, due to its training on larger data. A brief comparison can be seen as below:

	GPT-1	GPT-2/iGPT	GPT-3
No. of decoder blocks	12	48	96
Tokens	512	1024	2048
Corpus for training	Books	webText	WebText + books
Size of corpus	5GB	40GB	600GB
No. of parameters	117M	1.5B	175B

Table 2.1 – Comparison of GPT 1 vs GPT 2(iGPT) vs GPT 3

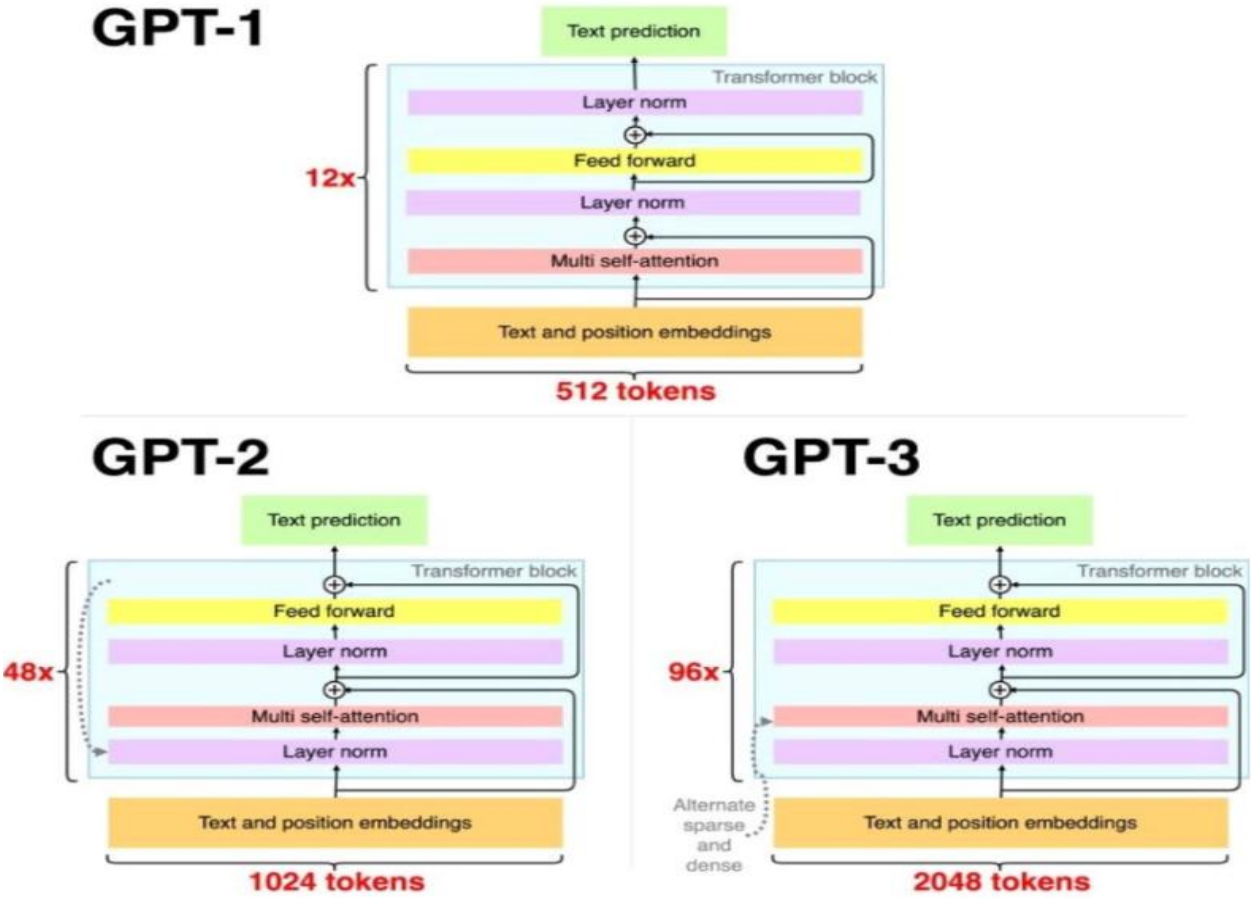

Fig 2.7– Comparison of GPT models architecture

Source: ResearchGate

2.4 Analysis of iGPT

The analysis of iGPT (imagined Generative Pretraining Transformer) involves evaluating its performance on various image generation and manipulation tasks, as well as understanding how the model generates images and what features it has learned.

- iGPT has been shown to generate high-quality images that are diverse, coherent, and visually pleasing. The model can generate images across a wide range of categories and styles, from natural scenes and animals to abstract shapes and textures. iGPT can also manipulate existing images, such as changing the colors, adding, or removing objects, or altering the background.
- One interesting aspect of iGPT's analysis is its ability to learn abstract concepts and features from unlabeled image data. For example, the model can generate images that exhibit symmetry, repetition, or texture patterns, even though these features are not explicitly labeled in the training data. This suggests that iGPT can learn higher-level representations of visual concepts that are useful for generating and manipulating images.
- Another aspect of iGPT's analysis is its comparison to other generative models, such as GANs and VAEs. iGPT has been shown to outperform these models on several image generation and manipulation tasks, including generating high-resolution images, controlling the style and content of images, and interpolating between different images smoothly.
- However, like all machine learning models, iGPT is not perfect and has limitations. For example, the model can sometimes generate unrealistic or ambiguous images, or fail to capture fine-grained details in the images. Moreover, the pre-training of iGPT requires large amounts of unlabeled data and computational resources, which can be a barrier for some applications.

1.Learning Objectives:

- **Pre-training:**
  1. Suppose we have an unlabeled dataset X containing high-dimensional data x = (x_1, ..., x_n). To model the density p(x), we can select a permutation π of the set [1, n] and apply an auto-regressive approach as follows:

$$\boldsymbol{p(x) = \prod_{i=1}^{n} p\left(x_{\pi_i} \mid x_{\pi_1}, \ldots x_{\pi_{i-1}}, \theta \right)}$$

  2. In the case of images, we typically choose the identity permutation πi = i for 1 ≤ i ≤ n, which is also referred to as raster order. The model is then trained to minimize the negative log-likelihood:

$$\boldsymbol{L_{AR} = E_{x \sim X}\left[-\log\left(p(x)\right)\right]}$$

3. The loss function utilized is similar to the masked language modeling approach used in BERT. This involves randomly sampling a sub-sequence M ⊂ [1, n], where each index i has an independent probability of 0.15 of being included in M.

$$L_{BERT} = E_{x\sim X} E_M \left[-\log \left(p\left(x_i \mid x_{[1,n]\setminus M} \right) \right) \right]$$

4. In the pre-training phase, we select either L_{AR} or L_{BERT} and aim to minimize the loss across our pre-training dataset.

- **Fine tuning:**
  In the fine-tuning stage, we utilized average pool n^L over the sequence dimension to obtain a d-dimensional vector of features per example. This vector was then used to learn a projection, which was subsequently employed to minimize the cross-entropy loss L_{CLF}. As a result, the overall objective function became:

$$f^L = \prec n_i^L \succ_i$$
$$L_{obj} = L_{GEN} + L_{CLF}$$

2. Dataset:

The MNIST database (Modified National Institute of Standards and Technology database) is a well-known collection of handwritten digits that is frequently used to train a variety of image processing systems. To create the dataset, the creators re-mixed samples from NIST's original datasets, as they believed that the original training dataset (which was taken from American Census Bureau employees) and testing dataset (which was taken from American high school students) were not ideal for machine learning experiments. Additionally, the original black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, resulting in grayscale levels.

The MNIST database consists of 60,000 training images and 10,000 testing images. Half of the training set and test set were derived from NIST's training dataset, while the other half was taken from NIST's testing dataset. The database's original creators keep a list of some of the methods that have been tested on it.

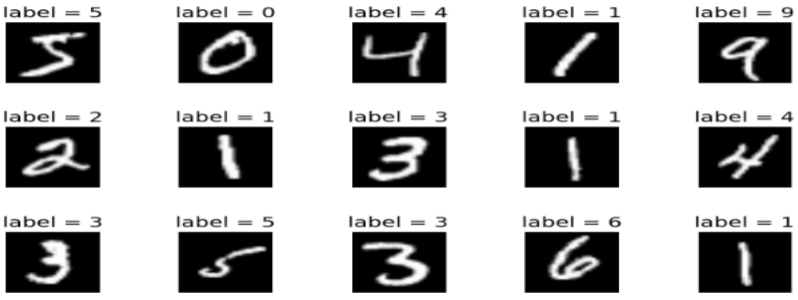

Fig 2.8: A sample from MNIST dataset

Source: ResearchGate

3. Implementation Details:

Image GPT is an extension of the GPT architecture designed for natural language processing. It is a generative model that can be used for image completion and image generation tasks. The architecture is similar to that of GPT, with the main difference being that the input to the network is an image instead of a sequence of text tokens. Here is a high-level overview of how Image GPT can be implemented:

- **Preprocessing**: The first step is to preprocess the input images. This can involve resizing the images to a standard size, normalizing the pixel values, and possibly augmenting the data with random rotations or translations.
- **Feature extraction**: Next, the images are fed through a pre-trained convolutional neural network (CNN) to extract features. The CNN typically consists of multiple convolutional and pooling layers that gradually reduce the spatial dimensions of the input image while increasing the number of channels.
- **Positional encoding**: To incorporate spatial information into the model, the features extracted by the CNN are combined with positional encoding. This involves adding learned positional embeddings to the feature vectors to represent the spatial position of each feature within the image.
- **Transformer network**: The combined feature vectors and positional embeddings are then fed through a multi-layer transformer network, similar to the one used in GPT. The transformer network is responsible for learning the distribution over possible completions or generated images.
- **Training**: The model is trained using maximum likelihood estimation (MLE) to minimize the negative log-likelihood of the training data. During training, the model is given partial input images and asked to predict the missing pixels or generate a new image from scratch.
- **Inference**: During inference, the model can be used to complete partial images or generate new images by sampling from the learned distribution. The generation process can be controlled by conditioning the model on additional inputs, such as text descriptions or attribute vectors.

Overall, implementing Image GPT involves combining a pre-trained CNN for feature extraction with a transformer network for learning the distribution over images. By leveraging the power of both convolutional and transformer architectures, Image GPT has shown impressive results in generating high-quality images.

4. Performance:

The performance of Image GPT can be measured in terms of the quality of the generated images and its ability to perform image completion tasks. Here are some of the key performance metrics for Image GPT:

1. Image quality: The quality of the generated images can be assessed by visual inspection, as well as metrics such as Inception Score (IS) and Fréchet Inception Distance (FID). These metrics measure the diversity and quality of the generated images by comparing them to a pre-trained classifier network.
2. Image completion accuracy: Image GPT can also be evaluated on its ability to perform image completion tasks, where it is given a partial image and asked to predict the missing pixels. The accuracy of these predictions can be measured by comparing them to the ground truth pixels.
3. Computational efficiency: Another important consideration for Image GPT is its computational efficiency, both during training and inference. This can be measured in terms of training time, memory usage, and inference time.

5. Applications:

iGPT has several potential applications in the field of computer vision and image processing. Here are some of the key applications of Image GPT:

1. **Image generation:** iGPT can be used to generate new, high-quality images from scratch. This has applications in fields such as art, design, and advertising, where novel and visually appealing images are highly valued.
2. **Image completion:** iGPT can also be used to complete partially observed images, where it is asked to predict the missing pixels based on the observed ones. This has applications in fields such as image editing, restoration, and medical imaging.
3. **Image synthesis:** iGPT can be used to synthesize images with specific attributes or properties, such as images of a certain style or genre. This has applications in fields such as entertainment, fashion, and advertising.
4. **Image manipulation:** iGPT can be used to manipulate existing images, such as by adding or removing objects, changing the background, or altering the lighting. This has applications in fields such as film, television, and advertising.
5. **Data augmentation:** iGPT can be used to generate new training examples for machine learning models, which can improve their performance on various tasks. This has applications in fields such as object detection, image classification, and segmentation.

6. Limitations:

While Image GPT has shown impressive performance in various tasks related to computer vision and image processing, it still has some limitations. Here are some of the key limitations of Image GPT:

1. **High computational requirements:** Image GPT requires large amounts of computing power and memory to train, which can make it difficult and expensive to use in some settings. This can also limit the size and complexity of the models that can be trained.

2. **Limited training data:** Image GPT relies on large amounts of training data to learn patterns and features in images. However, the availability of high-quality training data can be limited, particularly in domains where privacy and security concerns are paramount.
3. **Lack of interpretability:** Like other deep learning models, Image GPT can be difficult to interpret, which can limit its utility in some applications where interpretability is important. Understanding how Image GPT arrives at its predictions can be particularly challenging, given its complex architecture.
4. **Limited ability to handle certain types of images:** Image GPT can struggle to handle certain types of images, such as images with complex spatial relationships or images with multiple objects that overlap. This can limit its performance on certain tasks, particularly those that require a high degree of spatial reasoning.

2.4 Summary

Image GPT (iGPT) is a state-of-the-art deep learning model for image processing and computer vision tasks. It is a variant of the original GPT language model, which has achieved groundbreaking results in natural language processing. iGPT applies the same auto-regressive modeling approach to image processing, allowing it to generate highly accurate and detailed predictions of visual content. The iGPT model is pre-trained on a large dataset of images, allowing it to learn patterns and features that are useful for a wide range of computer vision tasks. The model uses a transformer-based architecture, which allows it to handle long-range dependencies and capture complex relationships between image features. The model is trained using a self-supervised learning approach, which means that it learns from the structure and content of the images themselves, without the need for explicit labels or annotations. One of the key advantages of iGPT is its ability to generate highly accurate and detailed predictions of visual content. For example, it has been shown to generate high-quality images from textual descriptions, perform accurate object detection, and generate realistic images from partial or incomplete inputs. This makes it a powerful tool for a wide range of applications, from image recognition and classification to content generation and image synthesis. Despite its impressive performance, iGPT does have some limitations. One of the main challenges of using iGPT is the high computational requirements needed to train and run the model. Training the model requires large amounts of computing power and memory, which can make it difficult and expensive to use in some settings. Additionally, the model can be challenging to interpret, particularly given its complex architecture, which can limit its utility in some applications where interpretability is important. Overall, iGPT represents a major breakthrough in the field of computer vision and image processing, offering a powerful and flexible tool for a wide range of applications. As research in this area continues to advance, it is likely that we will see even more impressive results from iGPT and other deep learning models for image processing and computer vision. In this paper, I will try to leverage iGPT for automatic image completion on MNIST dataset and will try to find ways to optimize the model performance.

3. SYSTEM REQUIREMENT SPECIFICATIONS

The requirements that are fulfilled by the project are discussed in detail, in this section. The perspective endeavored by the project, the features it is intended to offer, and the scope of the project is discussed in the upcoming sections. The Environment in which the product is executed, the assumptions and dependencies involved along with the testing requirements are discussed below. This chapter outlines the expectations of the project and the elements required to fulfil these expectations.

3.1 Project Scope

The population of the world is increasing with every passing day and due to the sedentary lifestyle (a lifestyle when someone spends six or more hours per day sitting or lying down, and they lack significant physical movement in their daily life), many lifestyle diseases are prevalent in young adults today like diabetes, cancer etc. Whereas the older generation suffer from health-related ailments like arthritis, dementia etc. The infants suffer from auto immune diseases, virus, and bacteria related diseases very frequently due to increase in pollution and overall lack of hygiene.

While the diseases are increasing, simultaneously the improvements and success of research in healthcare is also expanding, and people nowadays are getting more conscious due to increased knowledge and education about the symptoms. People prefer to get tested once a year and seeing the business prospect, many corporate held hospitals are providing lucrative offers to the multi-national companies for its employees as well as their families for a discounted price.

Since everything has its pros and cons, similarly with the awareness mentioned above, the number of health checkups done around the world are significantly increasing, and many life threatening diseases are getting diagnosed and prevented at nascent stages, but since the number of doctors and nurses are not increasing at a similar rate, it becomes too difficult for them to deal with this huge amount of data, because when the patient comes for a health check up for second year onwards, the healthcare professionals need to also look into their reports from the previous year. Now, imagine if a family of four, getting health checkup for five consecutive years, and for single member there is a report of approx. 30 pages per health checkup. For consecutive 5 years it will be 30*5 =150 pages and for a family of 4, it will be 600 pages of data, that needs to be checked before their sixth annual checkup. This number multiplied by $^{1}/_{20}{}^{th}$ of the world's population can easily become a nightmare for the limited number of healthcare professional that we have, which is 1 doctor per 250 patients in current scenario.

The only way to handle this issue with ease is to automate the process, which is where this paper comes in. Given a set of raw medical data (soft copy), the GPT-3 model will scan through

the data, process the data (keeping only relevant parts and removal of punctuations, emojis, stop words, special characters etc.) This data will be set as a prompt and made into a JSON file and will be sent for fine tuning. Once this process is completed, the model will be able to output Boolean answers, multiple answers very quickly for any question asked by a medical professional. This will save lots of time for the healthcare providers, and they will be able to quickly assess the patient's data before proceeding with the checkup. Additional benefit of this process will be that only the model needs the latest data of the current year, and it will not need to be trained in previous years data, as training had already been done on them. So, even with the increase in data, the performance of the model will not be challenged.

3.2 Project Perspective

This section explains the approach undertaken to fulfil the objectives of the project. As previously discussed in section 2, several approaches have already been tested for the image completion and the perks of the same are compared. After analyzing the various approaches available in the literature survey, which were fruitful yet could be more effective and efficient, I propose a perspective which uses the already available solutions with a humble yet valuable additional strategy, to make prompt generation more accurate but in lesser time.

3.2.1 Project Features

The major feature offered by the project is generation of output from large amount of medical data in a significantly less time. The project can only be executed on medical data, the accuracy might be challenged if done on data from other domains. Given a prompt with relevant context, the model will be able to generate response to the questions asked. The model needs to be trained whenever new data comes in, via a scheduler.

3.2.2 Assumptions, Dependencies and Constraints

The assumptions, dependencies and constraints set place in the development of the system are listed below:

1. The samples collected are from defacto MNIST dataset. Pytorch has processed this dataset in binary format, and this processed dataset has been used in this study.
2. The training requires a large and diverse set of data, and the processing of this large and diverse set needs reasonable processor speed and RAM. The cloud-based solutions such as Google Collab Pro cater to the needs of developing this model, while the traditional computing may not be very flexible.
3. In case, where the need is to predict newer image completion, the model needs to be trained with the corresponding dataset.

4. In this study, iGPT-Small model has been used, due to resource constraints. To train with iGPT-Medium, iGPT-Large and iGPT-XL, will require additional cost for higher resources.
5. This study has been conducted on black and white MNIST dataset. It can also be trained with colored MNIST dataset but that will require higher parameters and more resizing.
6. To increase the accuracy of the model, we trained iGPT-S with augmented data.
7. OpenAI's GPT-3 API is costly. Since this study deals with image datasets, hence the parameters will be more, as a result free limit will quickly exhaust and it will be costly, hence GPT-3 has been avoided.

3.3 Functional Requirements:

The basic functionalities expected of the project are listed below:

1. As a user, when I expect the model to do image completion on test image, I assume that the test image is of size 28x28 pixels, in grayscale.
2. As a user, when I expect the model to predict with higher accuracy, I assume that the model has been pre-trained with augmented data while testing.
3. As a user, when I expect the model to generate complete images, I assume that the test image is of the format png, jpg and jpeg only.
4. As a user, I expect the model to generate complete image of MNIST dataset only. I am aware that in case model is expected to predict newer images, I will need to train it with the corresponding dataset.
5. As a user, when I input a test image which is half filled, I expect the model to generate multiple completed images similar to the original image that was used while training the model.
6. As a user, when I input an augmented image (inverted, rotated etc) which is half completed, for testing, I expect the model to correctly generate completed image similar to the original dataset which was used during training.

3.4 Non-Functional Requirements:

Classification models often have multiple categorical outputs, which can make it difficult to identify specific instances of errors. Although most error measures calculate the total error in our model, they cannot show us where the model misclassifies certain categories more frequently than others. This is a limitation of standard accuracy measures.

In addition, if there is a significant class imbalance in the dataset, where one class has far more instances than the others, a model may appear to have a high accuracy score by simply predicting the majority class for all cases, while completely neglecting the minority classes. This is where confusion matrices come in handy.

A confusion matrix is a visual representation of the different outcomes of a classification problem, showing the predicted and actual results in a table format. This tool is particularly useful for understanding the nuances of a model's performance, especially in cases where class imbalances may be present.

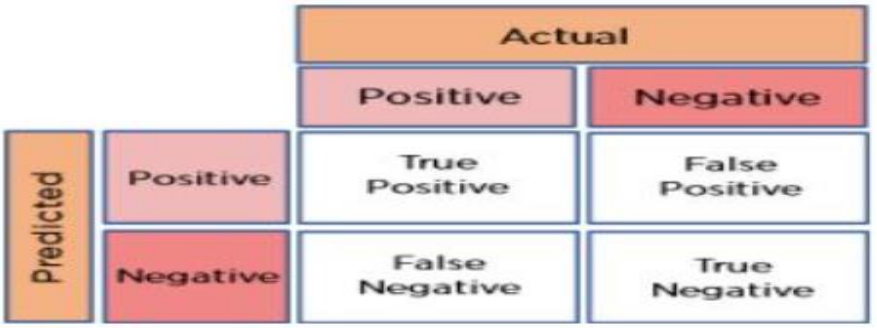

Fig 3.1 : A Confusion Matrix

Source: ResearchGate

- **True Positive (TP)**: The number of times our actual positive values are equal to the predicted positive. You predicted a positive value, and it is correct.
- **False Positive (FP)**: The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.
- **True Negative (TN)**: The number of times our actual negative values are equal to predicted negative values. You predicted a negative value, and it is actually negative.
- **False Negative (FN)**: The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.

It can be challenging to determine the performance of our model based solely on the confusion matrix. To gain a clearer understanding of our model's accuracy, we rely on a set of metrics, including:

1. **Accuracy:** The accuracy is used to find the portion of correctly classified values. It tells us how often our classifier is right. It is the sum of all true values divided by total values.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

2. **Precision:** Precision is used to calculate the model's ability to classify positive values correctly. It is the true positives divided by the total number of predicted positive values.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

3. **Recall:** It is used to calculate the model's ability to predict positive values. "How often does the model predict the correct positive values?". It is the true positives divided by the total number of actual positive values.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

4. **F1-Score:** It is the harmonic mean of Recall and Precision. It is useful when you need to take both Precision and Recall into account.

$$\text{F1-Score} = \frac{2 * \text{Precision*Recall}}{\text{Precision} + \text{Recall}}$$

3.5 System Requirements:

Initially the pre-processing, feature engineering and the model fitting was performed on a Laptop with the below mentioned hardware specification. As the size and dimensions of the dataset grew the model was moved to the Google Colab Pro Plus.

3.5.1 Hardware Requirements

The initial development was done on a MacBook with the following specifications:

RAM	16GB
Processor	Apple M1 chip, CPU @3.2GHz
Hard Disk	250GB
Number of CPUs	4 high performance “Firestorm” and 4 energy efficient “IceStorm “ cores = 8 core CPU
Number of GPUs	8 core GPU
Operating System	macOS Ventura 13.2.1

Table 3.1 – Hardware Requirements

3.5.2 Software Requirements

The software requirements of the project are tabulated below. During the later stages of development the system was developed on the Google Colab Pro Plus and the cloud resources used are as below:

Cloud environment	Google Colab Pro Plus
Development language	Python 3.8
Deep Learning Framework	Pytorch
Model Accelerator Framework	Pytorch Lightning
Third party Libraries	Pandas Numpy Sklearn Matplotlib yaml
Log monitoring Framework	Tensor Board

Table 3.2 – Software Requirements

4. PROPOSED METHODOLOGY

The study presented in this thesis describes a novel method of deriving the usage of a medicine when the medicine name is entered as input to GPT-2 . As described in the Chapter 2, there are studies in the past that targeted various problems in healthcare domain, but no system has tried to explore this problem statement. As a part of this thesis, I tried to implement the model with some minor modifications to improve the performance of the model.

The architecture of the system is described in 4.1. algorithm is described in 4.2, followed by a detailed description of the software design in 4.3.

4.1 System Architecture

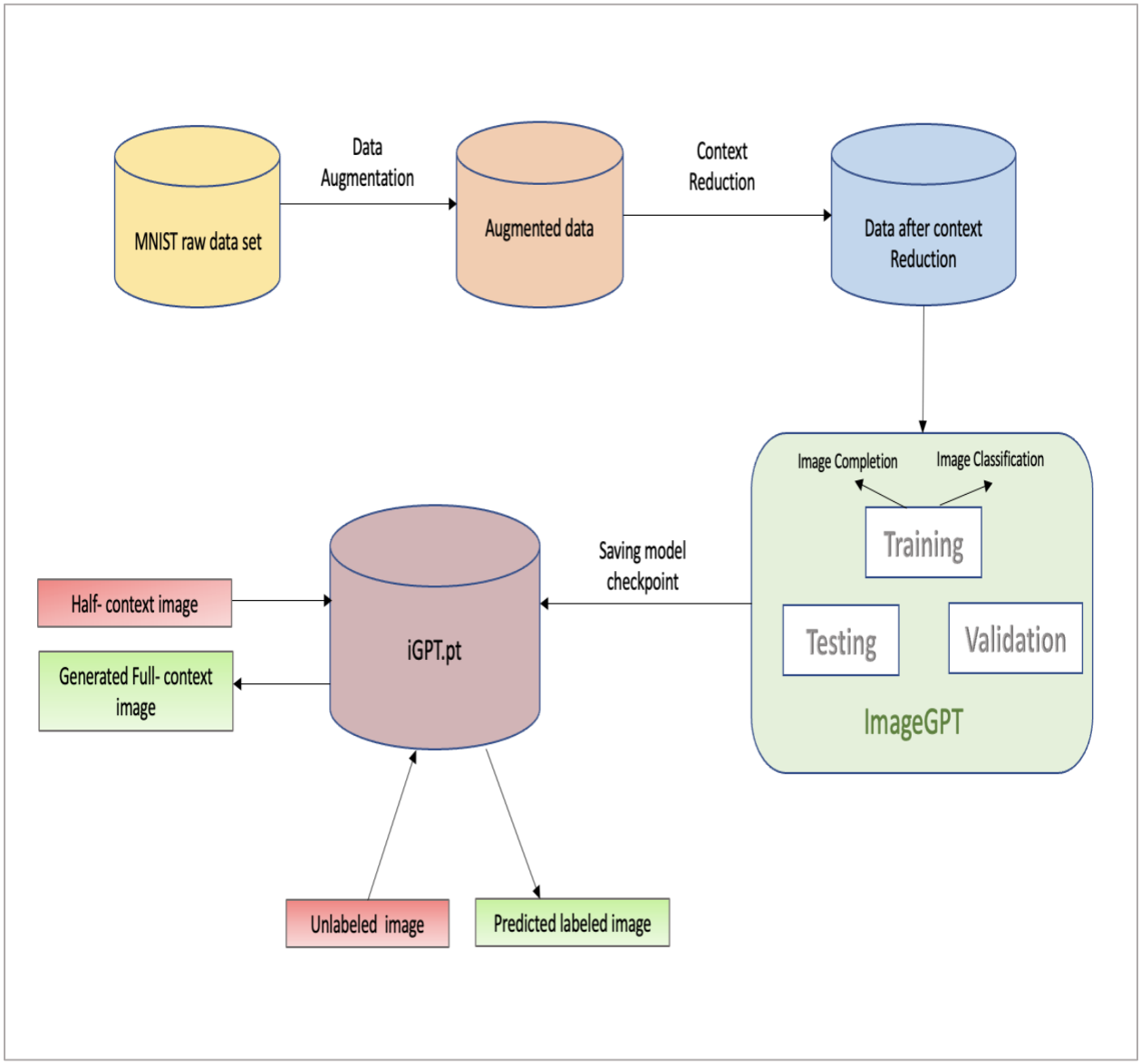

Fig 4.1– Architecture Diagram of proposed System

4.2 System Algorithm:

The algorithm of the system is as below:

1. *Download MNIST dataset from OpenAI blob store.*
2. *Send the data for augmentation (Random Crop, Horizontal Flip,etc)*
3. *Resize the images to 28x28.*
4. *Calculate centroid using MiniBatchKMeans*
5. *Save the centroid in Numpy File.*
6. *Send the centroid and config file to ImageGPT (GPT-2) model.*
7. *Create train, validation, test data set using DataLoader.*
8. *Send train, validation, and test data set for training (Pytorch lightning library is used to speed up training process).*
9. *Once training completes, send random cropped image from MNIST dataset.*
10. *Model predicts full context images.*

4.3 System Design:

The MNIST dataset is a widely used benchmark dataset in the field of machine learning and computer vision. It consists of a set of 70,000 handwritten digits that have been normalized and centered, each digit being a grayscale image of size 28x28 pixels. The dataset is split into a training set of 60,000 images and a test set of 10,000 images. The training set is used to train machine learning models, while the test set is used to evaluate their performance.

The MNIST dataset is often used as a starting point for learning how to build and train neural networks and has been used to benchmark many state-of-the-art machine learning algorithms. It is a well-known dataset in the machine learning community and has been used in a wide range of applications, including optical character recognition, image classification, and deep learning research. For this project we used torchvision.

Torchvision is a PyTorch library that provides 3 kinds of modules - a collection of datasets, transforms, and models, specifically designed for computer vision tasks. It is used to load and preprocess image data, and to define and train deep neural networks for image classification, segmentation, and detection. The “torchvision.datasets” module provides a set of standard datasets, including MNIST, CIFAR-10, CIFAR-100, and ImageNet. I downloaded the MNIST

dataset and loaded into PyTorch. 60000 training images were not enough to train the imageGPT model. To overcome that issue , image augmentation had been performed on training images.

The “torchvision.transforms” module provides a set of standard image transformations that can be applied to images, such as normalization, resizing, cropping, and flipping. These transformations are often used for data augmentation to improve the performance of deep neural networks. The transformed module from torchvision was used for image augmentation.

Context reduction was performed as the resolution of the image was high. After that image were sent for training, testing and validation to imageGPT model. Once the training completed the model was saved into disk in .pt file for future downstream task. Now if any half context input image is passed to the model it will generate the full image.

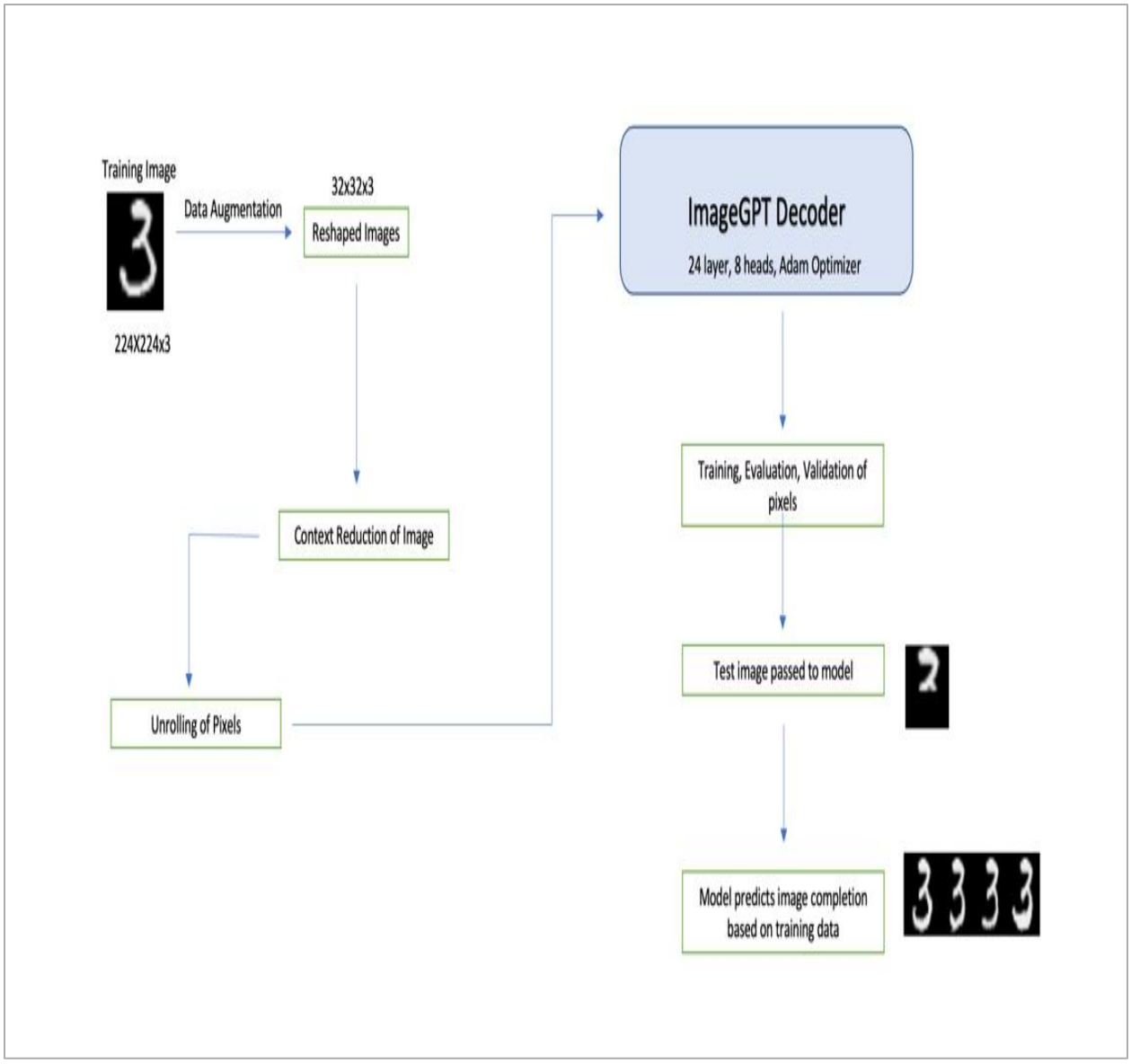

Fig 4.2– Block Diagram of the proposed System

5. IMPLEMENTATION DETAILS

This section describes the details of the implementation for the text generation part of the system.

5.1 Data Collection:

To train the imageGPT model on image data. It required a large number of images. It is quite standard to use MNIST data for any computer vision problem. Most of the computer vision solutions were benched marked on MNIST dataset. I followed the same for this project. There were few authentic sources which provided MNIST data in csv format but to train the model it required images. Pytorch has diverse set of libraries to support machine learning and deep learning algorithm, torchvision library is one of them. Torchvision library has dataset module. This module allows to download the MNIST data in binary format. Data was downloaded using that library. The downloaded image data was stored into disk for future purpose.

5.2 Data Augmentation:

Image augmentation is a technique used in computer vision and machine learning to increase the size and diversity of a training dataset by applying various image transformations to the original images. These transformations can include rotations, translations, flips, crops, zooms, color jittering, and other distortions. It increase the diversity and size of a training dataset, which can help improve the performance and generalization of machine learning models for computer vision tasks.

The goal of image augmentation is to introduce more variability in the dataset, which can help improve the robustness and generalization of the trained models. By training on a larger and more diverse set of images, models can learn to recognize patterns and features that are invariant to the specific transformations applied to the images.

For example, a model trained on a set of images of faces that have all been taken from the same angle may struggle to recognize faces taken from different angles. However, by applying random rotations and translations to the original images, the model can learn to recognize faces from any angle.

Image augmentation is particularly useful when the size of the training dataset is limited or when the dataset contains a limited variety of images. By artificially generating new images, image augmentation can help prevent overfitting and improve the generalization of the model to new data.

There are many libraries and frameworks that provide built-in support for image augmentation, including torchvision.transforms in PyTorch and ImageDataGenerator in Keras. These libraries provide a range of transformation functions and parameters that can be used to generate new images from the original dataset.

“torchvision.transforms” library was used for augmentation for MNITS dataset. As MNIST datasets was consists of grayscale images so few transformations were required for augmentations. *RandomCrop* with padding 4 pixel and *RandomHorizontalFlip* was used for that transformation. After transformations the train images were saved into disk. Before transformation, the size of the binary training images was 9.4 Mb. After augmentation size increased to 45 Mb.

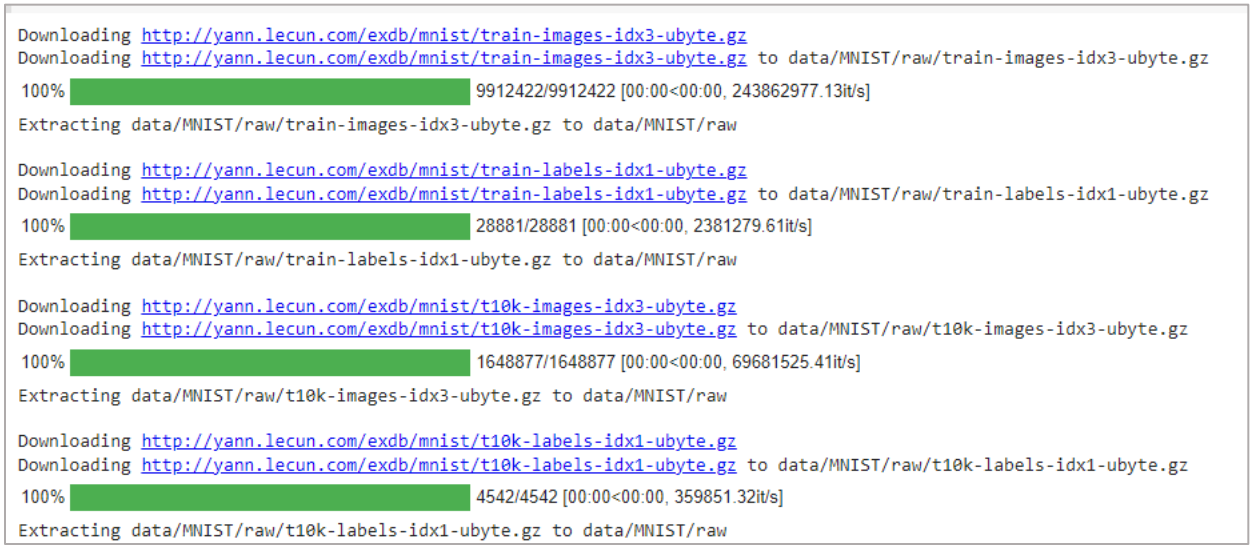

Fig 5.1– Extracting MNIST dataset via torchvision

5.3 View the augmented train data:

Here few examples are given for augmented data set. All the augmented images were reshaped to 28x28 pixel and flipped horizontally.

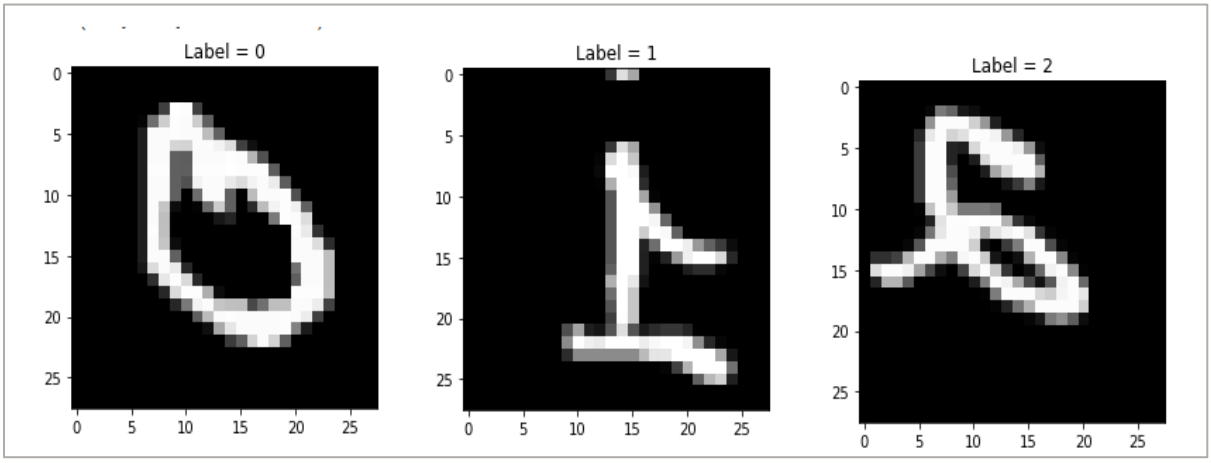

Fig 5.2– Augmented MNIST dataset (Label 0-2)

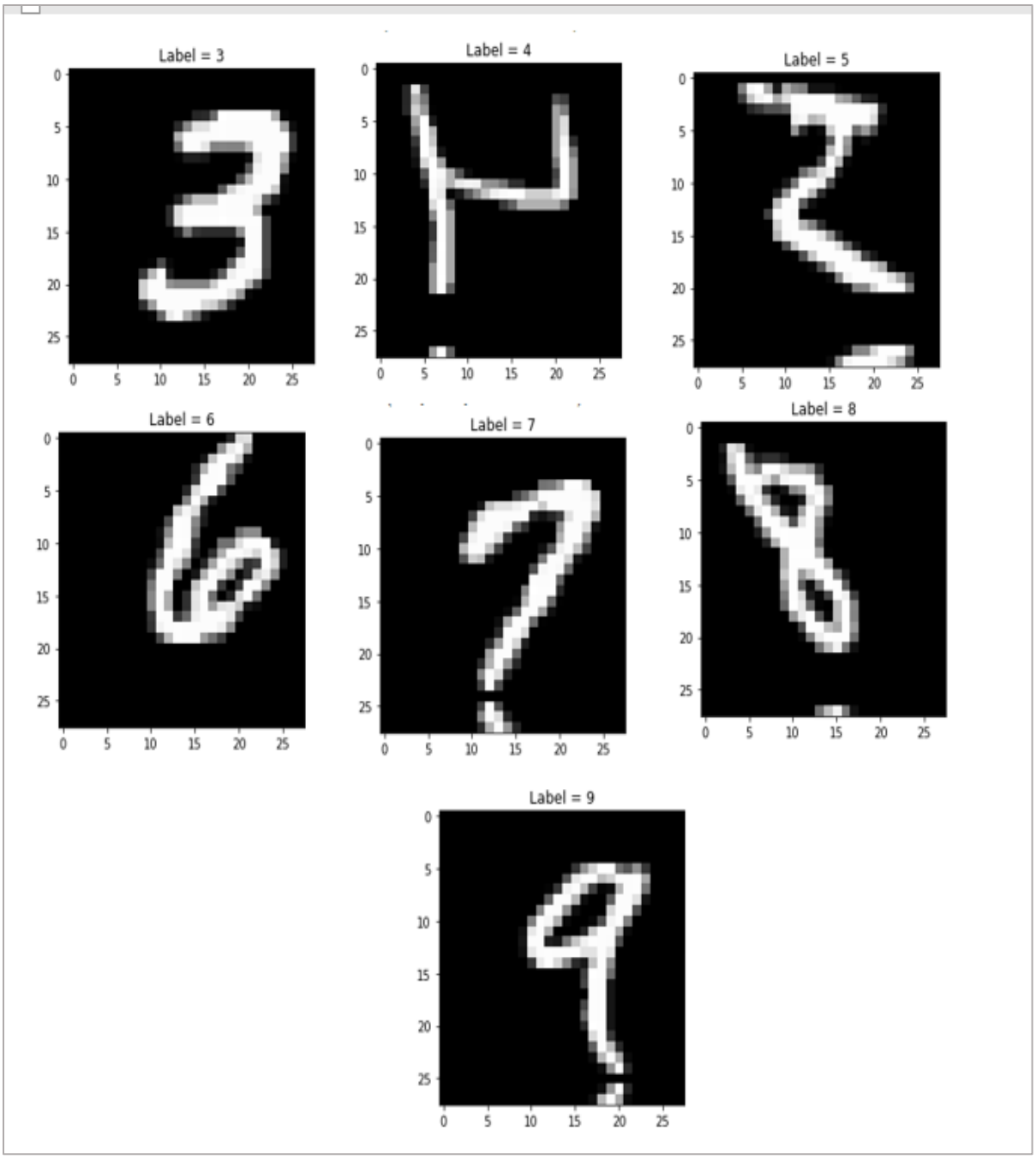

Fig 5.3– Augmented MNIST dataset (Label 3-9)

5.4 Context Reduction for images:

Context reduction is a technique used in this Image GPT (Generative Pre-trained Transformer) model, which is a variant of the popular GPT architecture designed for generating images. The goal of context reduction is to reduce the computational complexity of the model by reducing

the amount of context (i.e., the number of input tokens) that is processed at each layer of the network.

In this Image GPT model, the input to the network is a sequence of image patches, where each patch is a small square image of size 28x28 pixels. The patches are flattened and concatenated into a long sequence, which is then processed by a multi-layer transformer network to generate a sequence of output patches, which are combined to form the final image.

The transformer network consists of a series of transformer blocks, each of which consists of a self-attention mechanism and a feedforward network. The self-attention mechanism allows the network to attend to different parts of the input sequence, while the feedforward network applies a non-linear transformation to each input token.

However, as the input sequence becomes longer, the computational cost of processing each token increases, which can limit the size of the model and the size of the input images that can be processed.

To address this problem, this model uses context reduction to reduce the amount of context that is processed at each layer of the network. Specifically, the input sequence is divided into blocks of fixed length, and only a subset of the blocks are processed at each layer of the network. The output from each layer is then concatenated and used as the input to the next layer.

By processing only a subset of the input blocks at each layer, the model can reduce the computational cost of processing each token, while still allowing the network to attend to different parts of the input sequence. This can significantly increase the size of the model and the size of the input images that can be processed.

In addition to context reduction, the Image GPT model also uses a number of other techniques to improve the performance and efficiency of the network, including weight normalization, layer normalization, and a dynamic up sampling mechanism.

Context reduction technique was used for reducing the computational complexity of the Image GPT model. However, it was important to balance the amount of context reduction with the model's ability to attend to different parts of the input sequence, in order to achieve the best possible performance.

There were two options for me to use for centroid calculation one was *MiniBatchKMeans* and second one was *KMeans.* MiniBatchKMeans is an optimization of the KMeans clustering algorithm that is commonly used for unsupervised clustering of data. In MiniBatchKMeans, instead of computing the centroids of all data points in a batch, a smaller subset of the data, called a mini-batch, is used to update the centroids in each iteration. This makes MiniBatchKMeans faster and more scalable than the traditional KMeans algorithm, especially for large datasets.

The MiniBatchKMeans algorithm works by initializing a set of K centroids randomly from the data. Then, in each iteration, a random mini-batch of data points is selected and used to update the centroids. The algorithm computes the distance between each data point in the mini-batch and the current centroids and assigns each data point to the nearest centroid. The centroids are

then updated based on the mean of the data points assigned to each centroid. The algorithm continues to iterate, with new mini-batches selected at each iteration, until the centroids converge, or a maximum number of iterations is reached.

The advantage of MiniBatchKMeans is that it can be much faster than the traditional KMeans algorithm, especially for large datasets, since it updates the centroids using a smaller subset of the data in each iteration. This makes it more scalable and less memory-intensive, as it does not require loading the entire dataset into memory. However, the quality of the clusters may not be as good as KMeans since the algorithm does not use all the data points to compute the centroids.

MiniBatchKMeans was used for centroid calculation for this dataset with cluster size 16. After the centroid calculation the centroid was saved in .npy file for future purpose.

5.5 Showing the calculated centroid:

In clustering, the centroid is a representative point that defines the center of a cluster. It is typically the mean or the average position of all the data points in the cluster along each dimension. To find the centroid of a cluster, we first need to calculate the mean of the feature values for each dimension across all the points in the cluster. This gives us a vector of feature means, which serves as the centroid for the cluster. For example, suppose we have a dataset of two-dimensional points, and we want to cluster them into two groups using k-means clustering. After assigning the points to their respective clusters, we can calculate the centroid for each cluster as follows:

- For cluster 1, calculate the mean of the x and y values of all the points in the cluster.
- For cluster 2, calculate the mean of the x and y values of all the points in the cluster.

The resulting two mean vectors serve as the centroids for the two clusters. The centroid is an important concept in clustering, as it helps to define the location and shape of each cluster and can be used to assign new data points to the closest cluster based on their proximity to the centroid.

```
array([[4.6009950e-02],
       [0.0000000e+00],
       [9.0842378e-01],
       [6.4877909e-01],
       [1.9636802e-01],
       [9.5917130e-01],
       [8.4038848e-01],
       [1.0886785e-01],
       [9.9838632e-01],
       [5.3333485e-01],
       [7.5291753e-01],
       [1.4276293e-04],
       [9.9838412e-01],
       [3.0014792e-01],
       [9.9915224e-01],
       [4.1853124e-01]], dtype=float32)
```

Fig 5.4– Calculated centroid for 1 out of 16 clusters for label 1

As discussed in section 5.3, for each label between 0-9, 16 clusters were taken for each of them and then centroid was calculated for each of the clusters for every label as shown in Figure 5.4, and then they were arranged in ascending order with shortest centroid distance placed at the top. In base paper, 8 clusters were taken but to get better results I have taken 16 clusters in this paper.

5.6 Custom PyTorch implementation of the GPT2 language model:

GPT2 is a transformer-based language model developed by Open AI that uses unsupervised learning to generate human-like text but here it is used to generate pixel sequence. The custom GPT2 consists of two classes: Block and GPT2. The Block class represents a single transformer block, and the GPT2 class represents the entire GPT2 model. The Block class implements a single transformer block, which includes a multi-head attention layer, a feed-forward neural network, and layer normalization.

The forward method of the Block class applies layer normalization, performs multi-head attention, adds the attention output to the input, applies another layer normalization, passes the result through a feed-forward neural network, and adds the output of the feed-forward neural network to the input.

The GPT2 class initializes the start of sequence (sos) token, the token embeddings, and the position embeddings. It then applies the transformer block multiple times to the input sequence, concatenating the sos token and adding positional embeddings to the input before each transformer block.

After the final transformer block, it applies layer normalization and passes the result through a linear layer to obtain the output logits. If the "classify" argument is set to True, it also performs average pooling over the sequence, passes the result through another linear layer and returns both the classification logits and the generative logits.

5.7 Custom PyTorch implementation of the imageGPT model:

The PyTorch Lightning module used along with ImageGPT, that trains and fine-tunes a GPT-2 language model on the MNIST dataset. It generates images in the same way as text by treating each pixel as a token and flattening the image into a sequence of tokens, which the model then predicts one-by-one. The model can be used for both generative and discriminative tasks. The key components of the code are as follows:

a) The ImageGPT class inherits from the pl.LightningModule class, which provides a convenient and flexible way to define PyTorch models and training loops.

b) The *__init__* method initializes the GPT-2 model with the specified hyperparameters. It also loads a set of pre-defined MNIST image centroids, which are used to quantize the pixel values to a small number of discrete values. This reduces the size of the input vocabulary and allows the model to learn more efficiently.

c) The *configure_optimizers* method specifies the optimizer and learning rate scheduler for training the model. The optimizer is Adam with a fixed learning rate, and the learning rate scheduler is a linear warm-up followed by a linear decay.

d) The forward method of imageGPT simply calls the forward method of the GPT-2 model.

e) The *training_step* method performs a single forward and backward pass through the model on a batch of training data. It first quantizes the pixel values using the pre-defined centroids, then flattens the images into sequences of tokens, and finally passes the sequences through the GPT-2 model. If the model is being fine-tuned for classification, it also computes the classification loss and the generative loss and combines them into a joint loss. If the model is being used for generative tasks only, it simply computes the generative loss.The loss formula used for Image GPT, a variant of the GPT architecture designed for image generation tasks, is typically the negative log-likelihood of the target image given the model's predictions. Specifically, the loss can be expressed as:

$$\boldsymbol{L = -\log P\,(x \mid \theta)}$$

where L is the loss, x is the target image, and theta represents the model parameters. P(x | θ) is the probability of generating the target image x given the model's predictions, which is computed using the SoftMax function over the output distribution of the final layer of the model. In other words, the loss measures how well the model's predictions match the target image, with lower values indicating better performance. The goal of training is to minimize this loss over a large dataset of images.

f) The *validation_step* method performs a single forward pass through the model on a batch of validation data. It follows the same procedure as the *training_step* method but does not compute gradients. It returns the validation loss and, if the model is being fine-tuned for classification, it also returns the number of correct classifications predictions.

g) The *validation_epoch_end* method aggregates the validation loss and the number of correct classification predictions over all batches in the validation set and returns the average validation loss and validation accuracy.

h) The *test_step* and *test_epoch_end* methods are identical to the validation_step and validation_epoch_end methods, respectively. They are used to evaluate the model on a held-out test set after training is complete.

i) The train function loads a configuration file in YAML format, creates an instance of the ImageGPT model, and trains it is using the specified dataset and hyperparameters. If a pretrained model checkpoint is provided, it loads the checkpoint and modifies the model for fine-tuning.

j) The test function loads a pretrained model checkpoint and tests it on the specified dataset.

The imageGPT was trained against three different configuration which were small, medium, and large. While training checkpoints were created on 131000, 262000, 524000 and saved during training. After successfully training the saved model checkpoint was used for image generation and classification.

6. INTERMEDIATE RESULTS AND DISCUSSION

This chapter presents the intermediate results found for the procedures elaborated in the previous section, and also describes the datasets used for the project, and the evaluation metrics considered. The results presented here are intended only as a baseline and will be further worked upon to improve them in the next phase of the project.

6.1 Datasets:

The MNIST dataset is a widely used benchmark dataset in the field of machine learning and computer vision. It consists of a set of 70,000 images of handwritten digits, with each image represented as a 28x28 pixel grayscale image. The dataset is split into a training set of 60,000 images and a test set of 10,000 images, with each image labeled with its corresponding digit from 0 to 9.The MNIST dataset was created by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges in the late 1990s as a way to evaluate machine learning algorithms for recognizing handwritten digits. It has since become a standard benchmark dataset for evaluating the performance of various machine learning models, including deep neural networks. The MNIST dataset is widely used in the development of image recognition algorithms, particularly for digit recognition tasks. It has also been used as a benchmark dataset for evaluating generative models, such as variational autoencoders and generative adversarial networks.

Due to its simplicity and widespread availability, the MNIST dataset has become a popular starting point for many machines learning projects, particularly for those who are new to the field. However, it should be noted that while the MNIST dataset is a useful benchmark, it is not representative of all real-world image recognition tasks and may not be sufficient for evaluating the performance of more complex models.

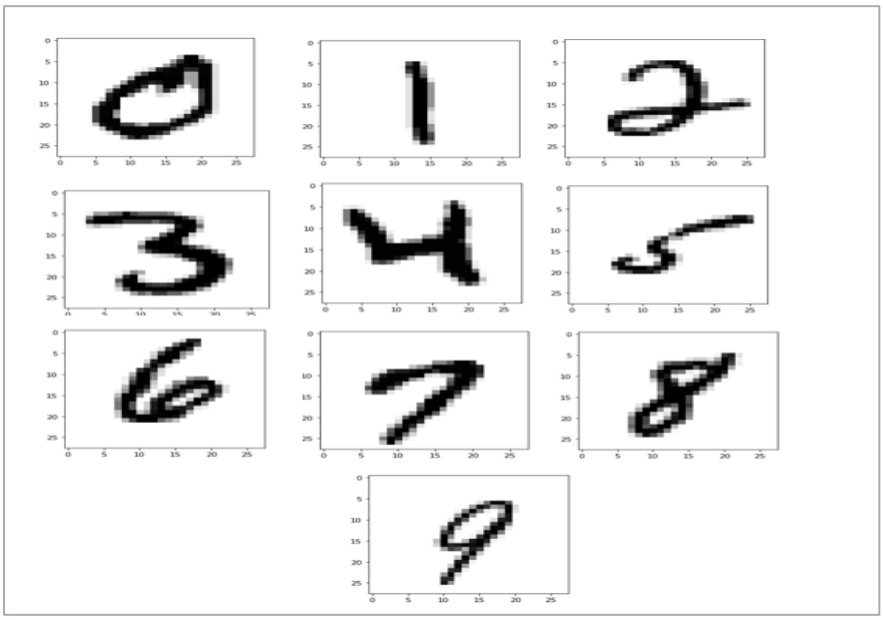

Fig 6.1 –Train dataset

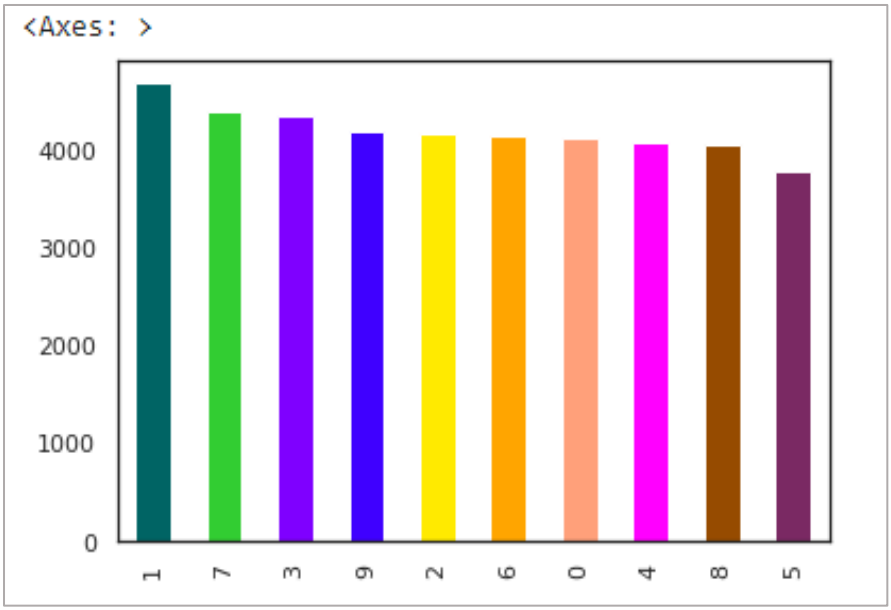

Fig 6.2 –Count of each label while training

6.2 Intermediate Results:

The imageGPT model has been trained for two tasks. One is image completion and second one is image classification. The image completion task for Image GPT involves generating plausible and coherent image completions for partial input images. In other words, given an image with missing parts, the model is trained to generate plausible and coherent completions for those missing parts. The image completion task is a challenging problem in computer vision, and has many practical applications, such as in photo editing and image restoration. Image GPT is well-suited for this task because of its ability to generate realistic and high-quality images based on learned patterns and correlations from large datasets.

To train the model for image completion, the partial input image is fed into the network, and the model generates a set of candidate completions, often in the form of a probability distribution over the possible pixel values. The output image is then selected from this distribution, typically using a sampling method such as top-k sampling or temperature-based sampling.

The quality of the generated image completions is evaluated using various metrics, such as visual similarity to the ground truth, perceptual quality, and overall coherence with the surrounding image context. The model is trained using techniques such as maximum likelihood estimation or adversarial training, with the goal of maximizing the likelihood of generating plausible and coherent completions for the partial input image. On the other hand image classification is a computer vision task that involves predicting the class or category of an image based on its content. In the context of Image GPT, the image classification task involves training a neural network model to classify images into one of several predefined categories based on

their visual features. To perform image classification using Image GPT, the model is first trained on a large dataset of labeled images. During training, the model learns to extract relevant features from the images and map those features to the corresponding class labels. The features are typically learned by passing the input image through a series of convolutional layers, which are designed to extract increasingly complex and abstract visual features from the image.

Once the model has been trained, it can be used to classify new images by passing them through the network and predicting the most likely class label. The accuracy of the model is typically evaluated using metrics such as classification accuracy or precision and recall. The image classification task is an important and widely studied problem in computer vision, with many practical applications, such as in image search, object detection, and content-based image retrieval. Image GPT is a powerful tool for image classification because it can capture complex and abstract visual features that are difficult to extract using traditional machine learning methods.

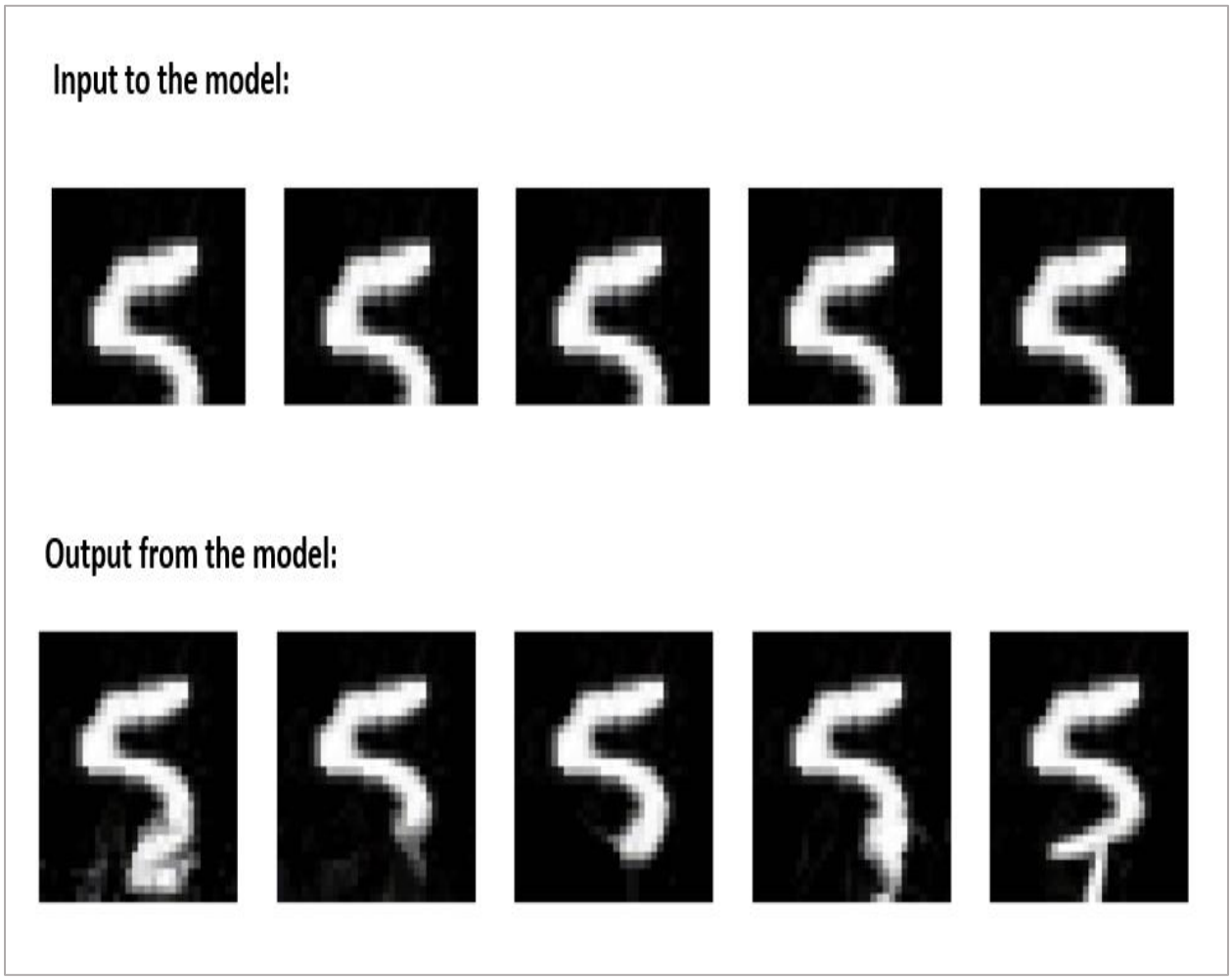

Fig 6.3 –Full context image prediction by the model

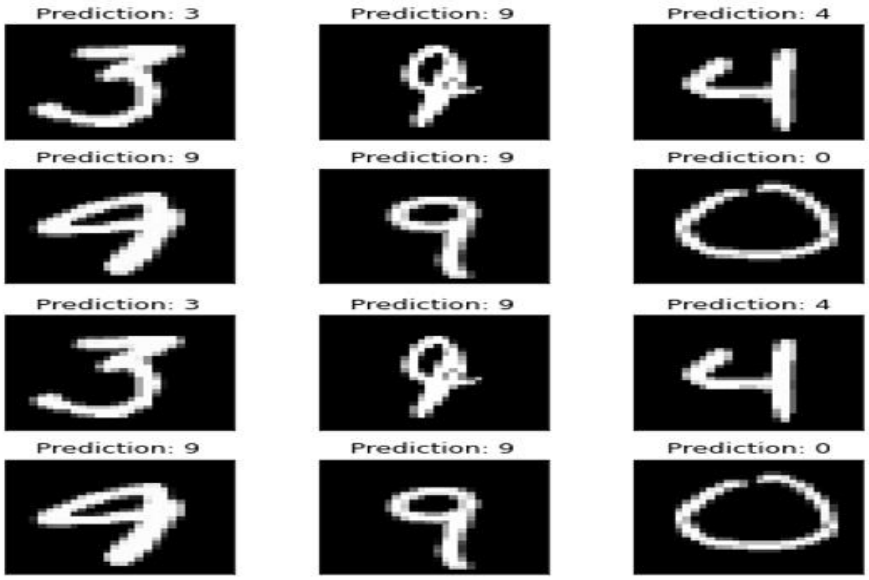

Fig 6.4 –Label prediction by the model

6.3 Model Performance Evaluation:

Model performance was measured using confusion matrix and kappa score. A confusion matrix is a table used to evaluate the performance of a machine learning model for a classification task. It is a matrix of size n x n, where n is the number of classes in the classification problem.

The rows of the matrix represent the actual class labels, while the columns represent the predicted class labels. Each cell in the matrix represents the number of instances that belong to a particular combination of actual and predicted class labels. The diagonal cells represent the correct predictions, while the off-diagonal cells represent the incorrect predictions. The confusion matrix is a useful tool for evaluating the performance of a classification model, as it provides a detailed breakdown of the types of errors made by the model. From the confusion matrix, several performance metrics can be derived, including accuracy, precision, recall, and F1-score. These metrics provide insights into the performance of the model and can be used to compare the performance of different models or parameter settings.

Whereas kappa score, also known as Cohen's kappa coefficient, is a statistic that measures the agreement between two raters, such as a machine learning model and a human annotator, in a categorical classification task. It is often used as a measure of inter-rater reliability. In the context of machine learning, the kappa score is used to evaluate the performance of a classification model in terms of how well it agrees with human annotators. The score ranges from -1 to 1, where a score of 1 indicates perfect agreement between the model and the human annotators, 0 indicates no agreement beyond chance, and -1 indicates complete

disagreement. The kappa score considers the possibility of agreement occurring by chance and adjusts the accuracy of the model accordingly. It is calculated by comparing the observed agreement between the model and the human annotators to the expected agreement that would occur by chance. The formula for calculating the kappa score is:

where the observed agreement is the proportion of instances where the model and the human annotators agree, and the expected agreement is the proportion of instances where agreement would be expected by chance.

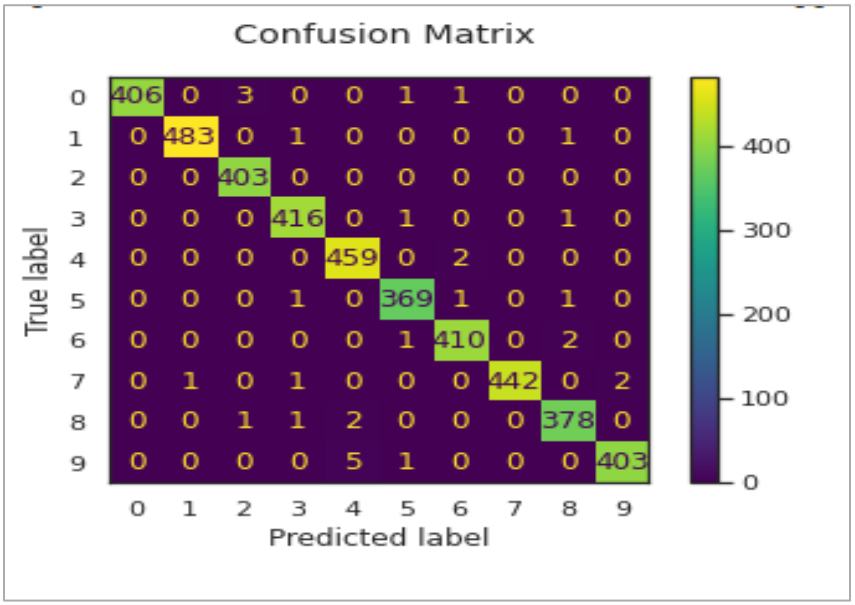

Fig 6.5 –Confusion matrix generated by the model

Here we can see that imageGPT model performs very well on all digits with few errors. However, it seems that model has some little troubles with the 4 digits, hey are misclassified as 9. Sometimes it is very difficult to catch the difference between 4 and 9 when curves are smoot.

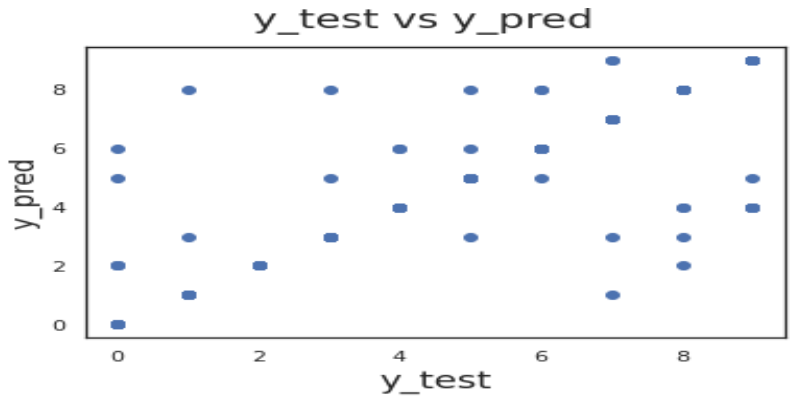

Fig 6.6 – Plot of Actual vs predicted label

From figure 6.5 we see that the number of points are far more diagonally compared to the rest of the plot. Hence we can say that the model has performed significantly well in prediction.

I calculated kappa score which came out to be 0.992. As the kappa score for the full model (with cut-off probability 0.5) is more than 0.99, it can be assumed that there is substantial agreement between the actual and predicted values.

6.4 Model Comparison:

In this section, I have compared the image completion loss, classification loss and classification accuracy with three different models which were imageGPT small, imageGPT medium and imageGPT large.

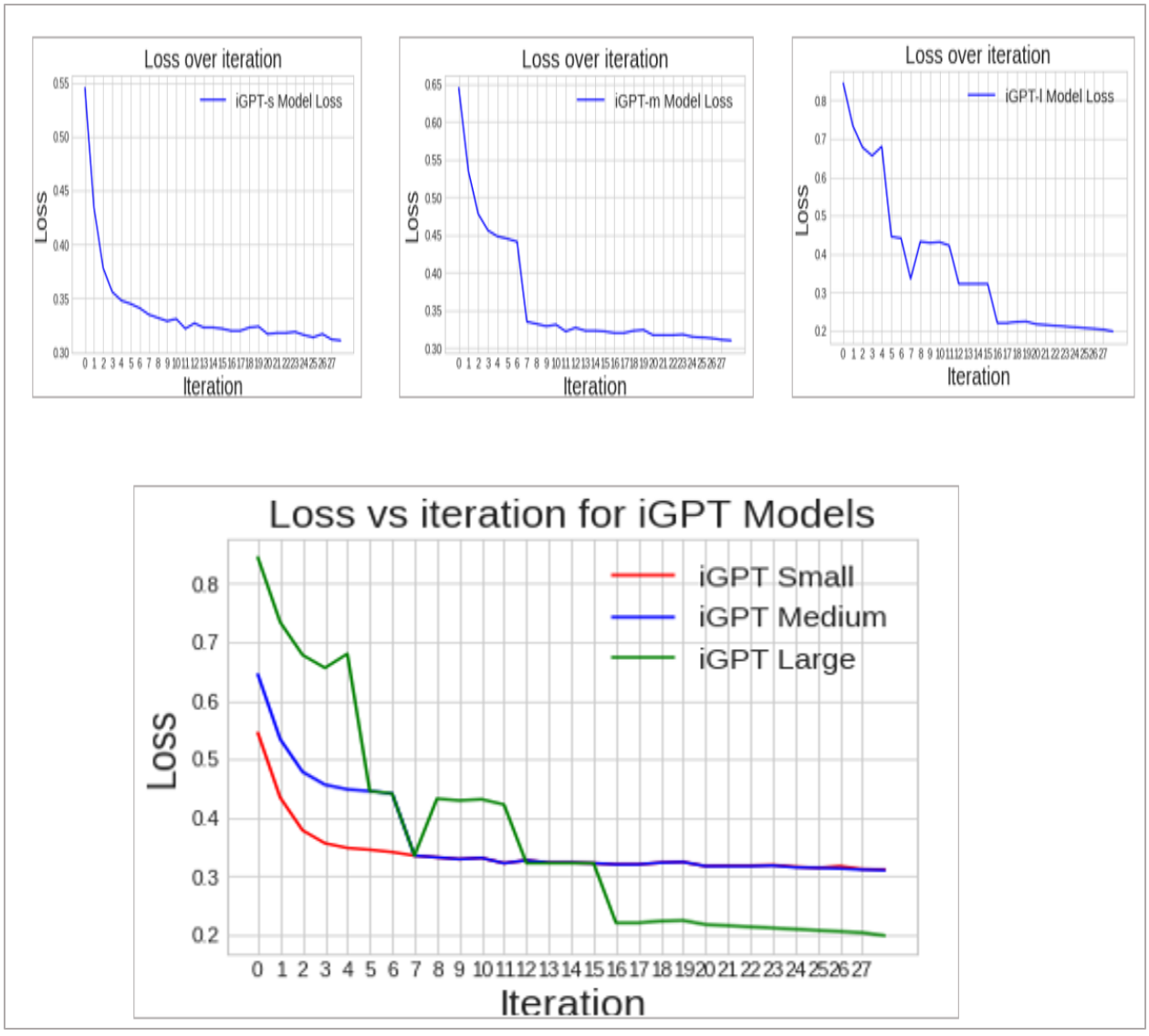

Fig 6.7 – Image completion loss for iGPT models

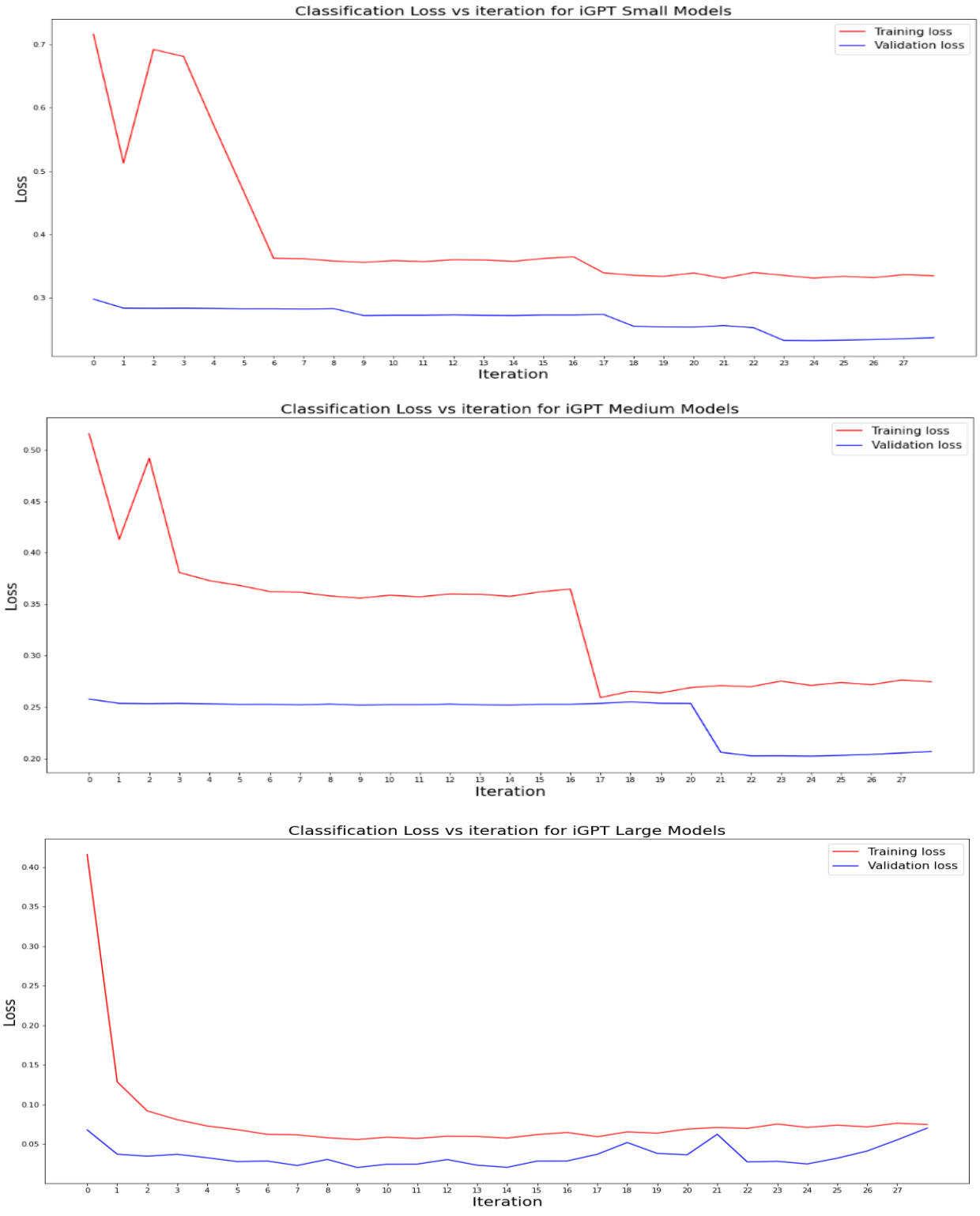

Fig 6.8 – Image classification loss for iGPT models

Fig 6.9)a– Image classification training loss for iGPT models

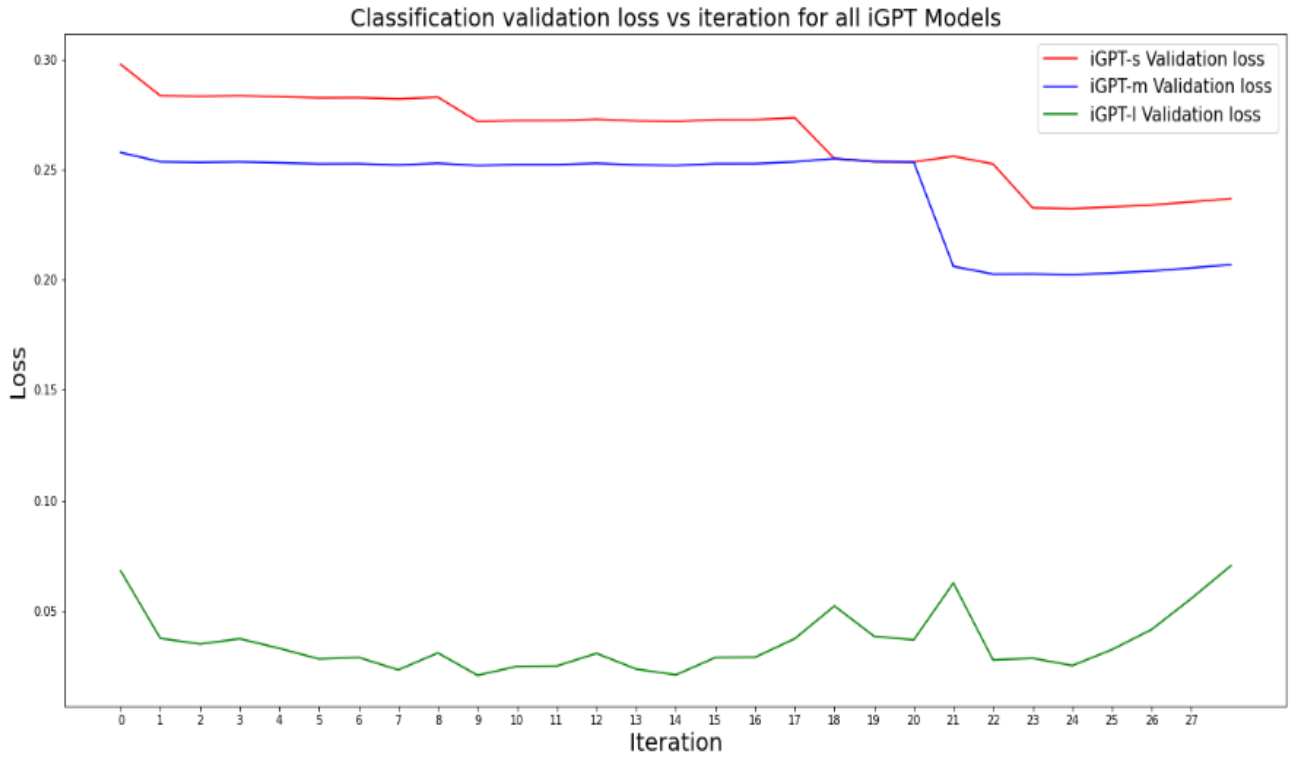

Fig 6.9)b – Image classification validation loss for iGPT models

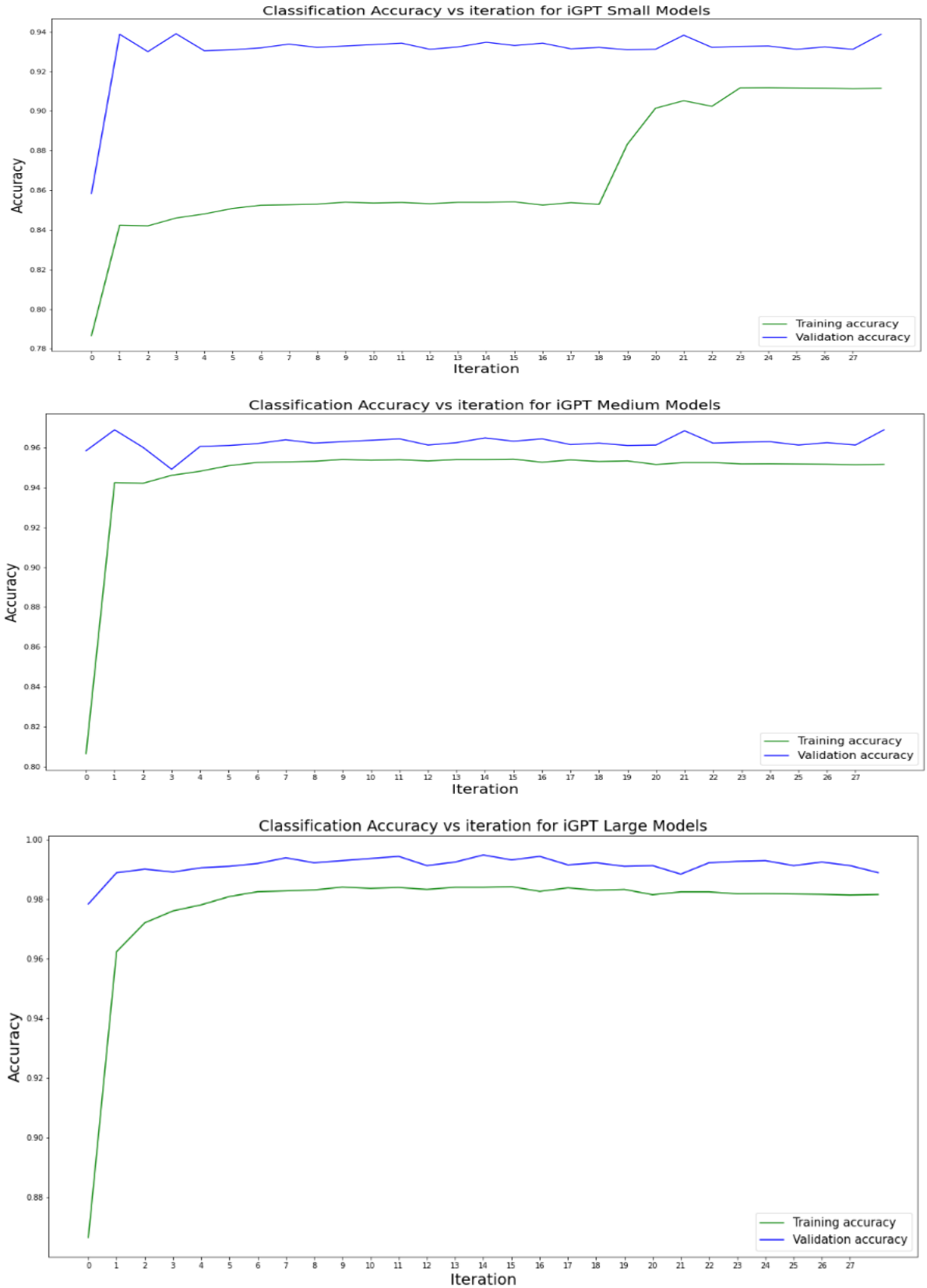

Fig 6.10– Image classification accuracy for iGPT models

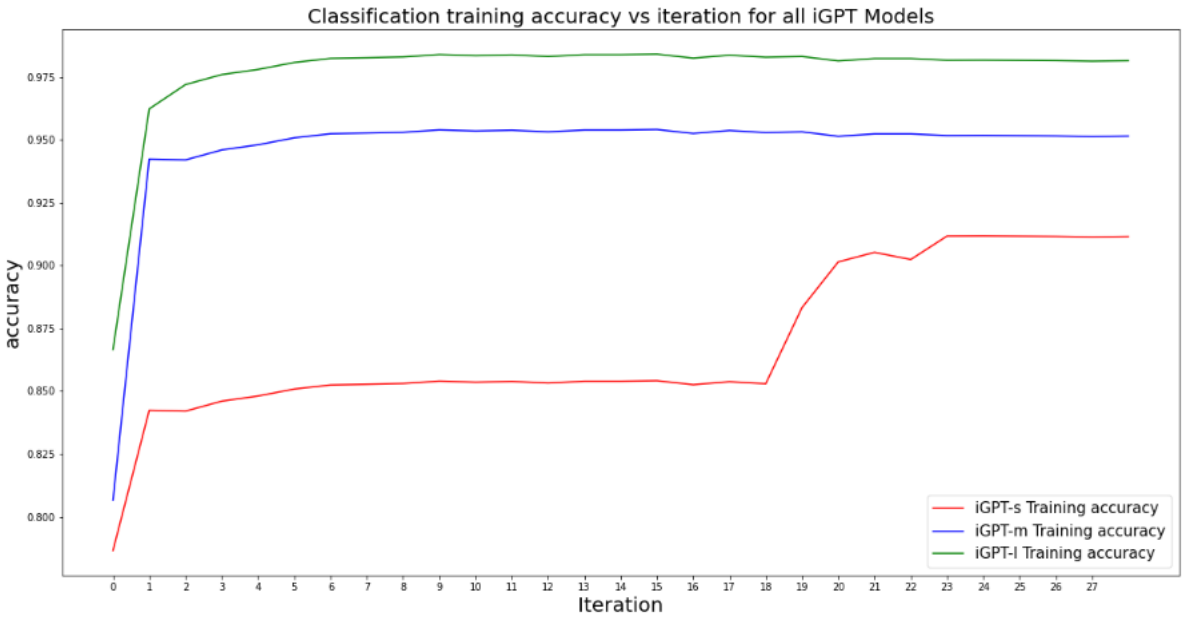

Fig 6.11)a– Image classification training accuracy for iGPT models

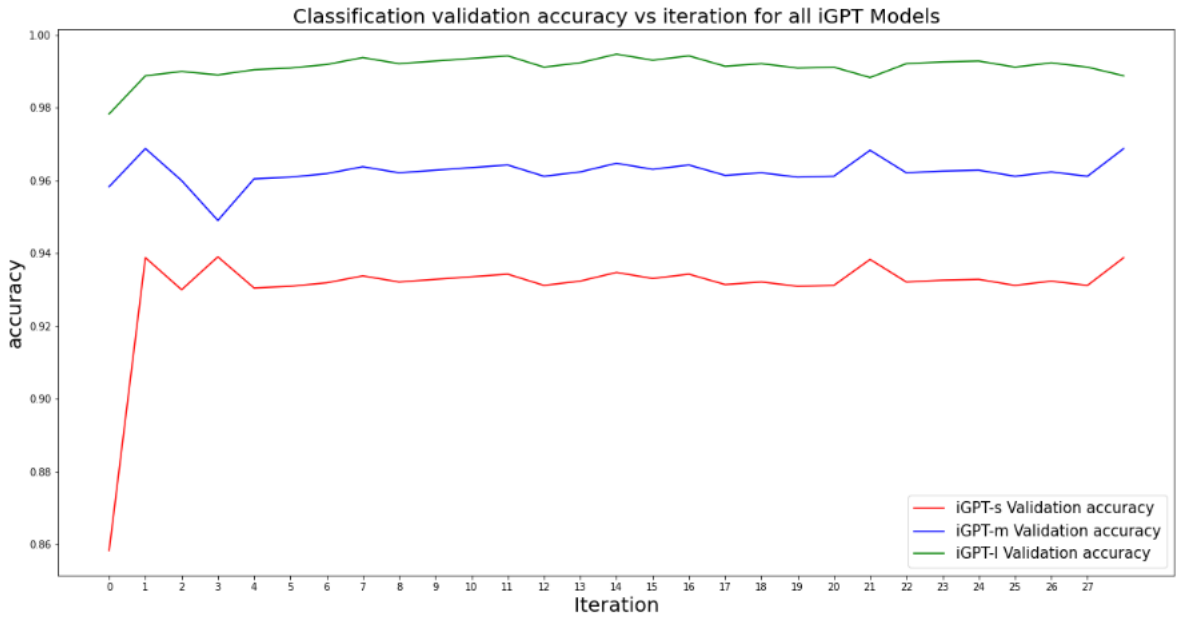

Fig 6.11)b– Image classification validation accuracy for iGPT models

From the above figures we can conclude that iGPT-large model has the highest accuracy and the lowest loss among all other iGPT models, this is because iGPT-large has the highest number of parameters and trained on more epochs.

7. CONCLUSION AND FUTURE SCOPE

Image GPT models represent an exciting development in the field of computer vision. These models use a combination of deep learning and natural language processing to generate captions, classify images, and even generate new images from textual input. In this section, we will evaluate the conclusion and future scope of Image GPT models.

7.1 Conclusion

Image GPT models have shown great promise in a variety of applications, including image captioning, image classification, and image synthesis. For example, the Image GPT model has been shown to generate highly accurate and natural language descriptions of images. These models can be used to automatically generate captions for images in large datasets, making it easier for humans to search and browse through these datasets. In addition to image captioning, Image GPT models can also be used for image classification tasks. For example, Image GPT models have been used to classify images of different types of animals, plants, and even human emotions. These models can help us better understand and categorize visual data, which can have applications in fields such as healthcare, robotics, and autonomous vehicles. Finally, Image GPT models can also be used for image synthesis. This involves generating new images based on textual input, such as a written description or a list of attributes. For example, an Image GPT model could be trained to generate realistic images of birds based on a textual description of their physical characteristics. This could have applications in fields such as fashion, architecture, and graphic design.

7.2 Future Scope

The future scope of Image GPT models is vast and exciting. One area of future development is in the generation of more colored images. The same model could be extended to include image completion as well as classification of colored images. The potential of iGPT-XL can also be evaluated with the help of higher computing resources. It has a scope to perform better than iGPT-L. In conclusion, Image GPT models represent a major advancement in the field of computer vision. These models have shown great promise in a variety of applications, and the future scope of these models is vast and exciting. As Image GPT models continue to develop and improve, we can expect to see even more innovative and useful applications of this technology.

References:

1. Chen, Mark and Radford, Alec and Child, Rewon and Wu, Jeffrey and Jun, Heewoo and Luan, David and Sutskever, Ilya. *Generative pretraining from pixels.* International conference on machine learning. Pages 1691-1703, PMLR, 2020.
2. Luo, Renqian, Sun, Liai , Xia, Yingce , Qin, Tao Zhang, Sheng , Poon, Hoifung, Liu and Tie-Yan. *BioGPT: generative pre-trained transformer for biomedical text generation and mining.* Briefings in Bioinformatics, Oxford Academic, 2022.
3. Su, Nigel, Yixuan and Collier. *Contrastive search is what you need for neural text generation.* arXiv preprint arXiv:2210.14140, 2022
4. Chang, Ernie and Shen, Xiaoyu and Zhu, Dawei and Demberg, Vera and Su, Hui. *Neural data-to-text generation with lm-based text augmentation.* arXiv preprint arXiv:2102.03556, 2021
5. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention is all you need.* corr abs/1706.03762 (2017). 2017.
6. Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. *Language models are unsupervised multitask learners*. 2018. URL- Language Models
7. Yang Liu and Mirella Lapata. *Text summarization with pretrained encoders.* arXiv preprint arXiv:1908.08345, 2019.
8. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. *Exploring the limits of transfer learning with a unified text-to-text transformer,* 2019.
9. Zhen Huang, Shiyi Xu, Minghao Hu, Xinyi Wang, Jinyan Qiu, Yongquan Fu, Yuncai Zhao, Yuxing Peng, and Changjian Wang*., "Recent trends in deep learning based open-domain textual question answering systems.,"* 2020.
10. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *Bert: Pre-training of deep bidirectional transformers for language understanding,* 2018.
11. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew*. Huggingface's transformers: State-of-the-art natural language processing,* 2019.
12. Derek Miller. *Leveraging bert for extractive text summarization on lectures*, 2019.
13. Dima Suleiman and Arafat Awajan. *Deep learning based abstractive text summarization: approaches, datasets, evaluation measures, and challenges. Mathematical problems in engineering,* 2020.
14. Philipp Hartl and Udo Kruschwitz. *Applying automatic text summarization for fake news detection.* arXiv preprint arXiv:2204.01841, pages 1–8, 2022.
15. Abigail See, Peter J Liu, and Christopher D Manning. *Get to the point: Summarization with pointer-generator networks.* arXiv preprint arXiv:1704.04368, 2017
16. Dmitrii Aksenov, Julián Moreno-Schneider, Peter Bourgonje, Robert Schwarzenberg, Leonhard Hennig, and Georg Rehm. *Abstractive text summarization based on language model conditioning and locality modeling.* arXiv preprint arXiv:2003.13027, 2020.
17. Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. *Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter*, 2019
18. Shakshi Sharma, Ekanshi Agrawal, Rajesh Sharma, and Anwitaman Datta. *Facov: Covid-19 viral news and rumors fact-check articles dataset. In Proceedings of the International AAAI Conference on Web and Social Media,* volume 16, pages 1312–1321, 2022.
19. Ruixuan Zhang, Zhuoyu Wei, Yu Shi, and Yining Chen*. Bert-al: Bert for arbitrarily long document understanding,* 2019.

20. Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Kinney, Ziyang Liu, William Merrill, Paul Mooney, Dewey Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. *Cord-19: The covid-19 open research dataset,* 2020
21. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural machine translation by jointly learning to align and translate*. arXiv preprint arXiv:1409.0473, 2014.
22. Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.
23. Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. *Transformer-xl: Attentive language models beyond a fixed-length context.* arXiv preprint arXiv:1901.02860, 2019.
24. Akbar Karimi, Leonardo Rossi, and Andrea Prati. *Adversarial training for aspect-based sentiment analysis with bert*. pages 8797–8803, 2021.
25. Mohamed El Ghaly Beheitt and Moez Ben Haj Hmida. *Automatic arabic poem generation with gpt-2*. pages 366–374, 2022.
26. Juan-Manuel Torres-Moreno. *Automatic Text Summarization*. Wiley, 2014.
27. Hritvik Gupta and Mayank Patel*. Method of text summarization using lsa and sentence based topic modelling with bert.* pages 511–517, 2021.
28. Francisca Adoma Acheampong, Henry Nunoo-Mensah, and Wenyu Chen. *Transformer models for text-based emotion detection: a review of bert-based approaches.* Artificial Intelligence Review, 54(8):5789–5829, 2021.
29. Stephane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. *On the use of bert for neural machine translation.* arXiv preprint arXiv:1909.12744, 2019.
30. Neel Kanwal and Giuseppe Rizzo. *Attention-based clinical note summarization. In Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing,* pages 813–820, 2022.
31. Milad Moradi, Georg Dorffner, and Matthias Samwald. *Deep contextualized embeddings for quantifying the informative content in biomedical text summarization*. Computer methods and programs in biomedicine, 184:105117, 2020.
32. Danqing Wang, Pengfei Liu, Ming Zhong, Jie Fu, Xipeng Qiu, and Xuanjing Huang. *Exploring domain shift in extractive text summarization.* arXiv preprint arXiv:1908.11664, 2019.
33. Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: *Evaluating generated text as text generation*. Advances in Neural Information Processing Systems, 34:27263–27277, 2021.
34. Anna Glazkova and Dmitry Morozov. *Applying transformer-based text summarization for keyphrase generation*. arXiv preprint arXiv:2209.03791, 2022.
35. Bhrugesh Joshi, Vishvajit Bakarola, Parth Shah, and Ramar Krishnamurthy. *Deepmine-natural language processing based automatic literature mining and research summarization for early-stage comprehension in pandemic situations specifically for covid-19*. bioRxiv, 2020.
36. Shakshi Sharma, Ekanshi Agrawal, Rajesh Sharma, and Anwitaman Datta. Facov: *Covid-19 viral news and rumors fact-check articles dataset.* In Proceedings of the International AAAI Conference on Web and Social Media, volume 16, pages 1312–1321, 2022.
37. Soham Poddar, Azlaan Mustafa Samad, Rajdeep Mukherjee, Niloy Ganguly, and Saptarshi Ghosh. *Caves: A dataset to facilitate explainable classification and summarization of concerns towards covid vaccines*. arXiv preprint arXiv:2204.13746, 2022.
38. Niculescu, Mihai Alexandru and Ruseti, Stefan and Dascalu, Mihai. *RoGPT2: Romanian GPT2 for Text Generation.* 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), IEEE,2021

39. Pascual, Damian and Egressy, Beni and Meister, Clara and Cotterell, Ryan and Wattenhofer, Roger, *A plug-and-play method for controlled text generation,* arXiv preprint arXiv:2109.09707, 2021.
40. Deng, Shumin and Zhang, Ningyu and Yang, Jiacheng and Ye, Hongbin and Tan, Chuanqi and Chen, Mosha and Huang, Songfang and Huang, Fei and Chen, Huajun, *LOGEN: few-shot logical knowledge-conditioned text generation with self-training,* arXiv preprint arXiv:2112.01404, 2021.
41. Bai, He and Shi, Peng and Lin, Jimmy and Tan, Luchen and Xiong, Kun and Gao, Wen and Liu, Jie and Li, Ming. *Semantics of the unwritten: The effect of end of paragraph and sequence tokens on text generation with GPT2.* arXiv preprint arXiv:2004.02251, 2020
42. Chan, Alvin and Ong, Yew-Soon and Pung, Bill and Zhang, Aston and Fu, Jie. *Cocon: A self-supervised approach for controlled text generation.* arXiv preprint arXiv:2006.03535, 2020
43. Gong, Heng and Sun, Yawei and Feng, Xiaocheng and Qin, Bing and Bi, Wei and Liu, Xiaojiang and Liu, Ting. *Proceedings of the 28th International Conference on Computational Linguistics*. Pages - 1978—1988, 2020.
44. Mager, Manuel and Astudillo, Ram{\'o}n Fernandez and Naseem, Tahira and Sultan, Md Arafat and Lee, Young-Suk and Florian, Radu and Roukos, Salim. *GPT-too: A language-model-first approach for AMR-to-text generation.* arXiv preprint arXiv:2005.09123, 2020.
45. Groenwold, Sophie and Ou, Lily and Parekh, Aesha and Honnavalli, Samhita and Levy, Sharon and Mirza, Diba and Wang, William Yang. *Investigating African-American Vernacular English in transformer-based text generation.* arXiv preprint arXiv:2010.02510, 2020.
46. McCoy, R Thomas and Smolensky, Paul and Linzen, Tal and Gao, Jianfeng and Celikyilmaz, Asli. *How much do language models copy from their training data? evaluating linguistic novelty in text generation using raven*, arXiv preprint arXiv:2111.09509, 2021.
47. Rosati and Domenic. *SynSciPass: detecting appropriate uses of scientific text generation*. arXiv preprint arXiv:2209.03742, 2022.
48. Fang and Jingwu. *An Application of Customized GPT-2 Text Generator for Modern Content Creators.* University of California, Los Angeles, 2021.
49. Karpinska, Marzena and Akoury, Nader and Iyyer, Mohit. *The perils of using mechanical turk to evaluate open-ended text generation,* arXiv preprint arXiv:2109.06835, 2021.
50. Montesinos, Dimas Munoz. *Modern Methods for Text Generation.* arXiv preprint arXiv: 2009.04968, 2021.