GAN-Based Telugu Character Generation

Mini Project Report

Bachelor

in

Computer Science

By

S190200

S190689

S190698

S190862

Rajiv Gandhi University Of Knowledge Technologies

Campus-3 , Srikakulam -521201

Andhra Pradesh, India

Abstract

It is hard to build effective machine learning models for Telugu language processing due to the lack of huge datasets including Telugu characters. We propose a method for creating huge datasets from pre-existing character datasets in order to address this problem. We employ sophisticated data augmentation methods on the original character images, like noise and scaling, to create new versions. Additionally, the dataset is expanded through the application of Generative Adversarial Networks (GANs) to produce realistic Telugu characters. Machine learning models perform better as a result of this method's increased volume and diversity of data. Assessments indicate that the accuracy and robustness of character recognition systems are enhanced by these enriched datasets. Additionally, by providing comprehensive data for training and testing, we support the advancement of Telugu language technology. Better Telugu character recognition abilities could be used for more than only machine learning applications; they could also be used to digitise and preserve historical documents, manuscripts, and other works of literature written in Telugu script. This improves access to Telugu heritage items and makes a significant contribution to cultural preservation projects.

Contents

Chapter 1

Introduction

1.1 About the Project

This project tackles the challenge of limited Telugu character datasets by employing advanced data augmentation techniques and Generative Adversarial Networks (GANs) to generate new data. By incorporating scaling, noise, and GAN-generated realistic characters, we significantly expand the dataset. This enhanced dataset improves the accuracy and robustness of character recognition algorithms, benefiting machine learning models. Additionally, the project supports Telugu language technology advancement and aids in the digitization and preservation of cultural literature and historical documents.

1.2 Application

- The project supports the digitization and preservation of historical Telugu manuscripts and literature, aiding cultural preservation.
- Assistive technologies for visually impaired users are enhanced with better text-to-speech and braille conversion systems. Automated document processing in libraries, archives, and government institutions is streamlined, improving search and retrieval of Telugu texts.
- In Optical Character Recognition (OCR), it enhances systems to accurately digitize

printed and handwritten Telugu documents, improving accessibility and searchability.

- In Natural Language Processing (NLP), the enriched datasets boost tasks like translation, sentiment analysis, and text-to-speech, benefiting chatbots and virtual assistants.
- Educational Tools also gain from improved language learning apps for teaching Telugu script.
- Lastly, content creation and media benefit from advanced tools for generating and editing Telugu text and better subtitling and transcription services for Telugu audio and video content.

1.3 Motivation Towards the Project

This effort is motivated by the lack of extensive and varied datasets for Telugu characters, which is impeding the creation of effective machine learning models and Telugu language processing applications. This study tries to address these restrictions by creating large datasets using Generative Adversarial Networks (GANs) and sophisticated augmentation techniques. The improved datasets should increase the Telugu character recognition algorithms' robustness and accuracy, which will be advantageous for a number of industries including automated document processing, natural language processing, education, assistive technology, and OCR. In the end, this project helps to advance Telugu language technology, preserve cultural legacy, and encourage technological innovation in linguistic variety.

1.4 Problem Statement

Using Generative Adversarial Networks (GANs) to generate a large dataset of Telugu characters in order to enhance the preservation and use of old manuscripts.

Chapter 2

Approach to the Project

2.1 Project Description

This study addresses a critical challenge in Telugu language processing: the scarcity of large, diverse datasets containing Telugu characters, which hinders the development of efficient machine learning models. To overcome this obstacle, we propose an innovative approach using Generative Adversarial Networks (GANs) to generate extensive datasets of Telugu characters.

Our initial attempts to create a dataset by assigning Unicode values to each Telugu character resulted in poor image outputs during GAN training. To resolve this issue, we adopted a more sophisticated method utilizing an extensive .pkl dataset. This dataset comprised 38,915 images, with multiple class labels, each containing approximately 100 samples. The choice of .pkl files was strategic, as they offer significant technical advantages for machine learning applications. These files efficiently store and retrieve complex data structures like nested dictionaries and multidimensional arrays, which are essential for managing diverse datasets of Telugu character images. Moreover, .pkl files support compression techniques that optimize storage capacity without compromising data integrity, making them ideal for handling large volumes of image data in memory-constrained environments.

We implemented several key parameters to optimize the GAN's performance:

- Epochs: We increased the number of training epochs from an initial 100 to 2000. This substantial increase allowed for more extensive and refined learning, significantly enhancing the quality of generated images.

- ncritic=5: This parameter ensures that the discriminator is updated five times for each generator update, promoting more stable training by maintaining a balance between the two networks.
- Learning rate=0.00005: This relatively low learning rate allows for gradual and stable convergence, preventing overshooting optimal parameter values and ensuring smooth training progression.
- Batch Size=256: This batch size strikes a balance between computational efficiency and model stability, allowing for effective gradient updates without overwhelming system memory.
- Buffer Size=60000: A large buffer size improves the shuffling of training data, reducing potential biases and overfitting by ensuring a more random sampling of the dataset during training.
- noise dimension=100: This 100-dimensional noise vector provides sufficient complexity for the generator to create diverse and detailed Telugu characters.
- clip value=0.01: This parameter prevents the discriminator from becoming too powerful by clipping its weights, maintaining a balance in the adversarial training process.

In terms of activation functions, we employed Sigmoid activation in the discriminator's output layer and LeakyReLU activation in its hidden layers during training. For the generator model, we used the Tanh activation function to produce Telugu character images. These choices of activation functions were crucial in enabling the discriminator to effectively distinguish between real and generated images while allowing the generator to create high-quality outputs.

Our goal in fine-tuning these parameters and processes was to enhance the accuracy and reliability of Telugu character recognition systems. Beyond improving machine learning applications, this project contributes significantly to cultural heritage preservation. By facilitating the digitization and preservation of old Telugu books, manuscripts, and documents, we're making Telugu historical pieces more accessible and supporting broader initiatives in cultural heritage conservation.

The success of this approach not only advances the field of Telugu language processing but also demonstrates the potential of GANs in addressing similar challenges in other languages with limited digital resources. This research opens up new possibilities for preserving and studying linguistic and cultural heritage through advanced machine learning techniques.

2.2 Data Set

The dataset is kept in a format called.pkl.The dataset used for this study consists of 38,915 pictures of Telugu characters arranged into 397 different labels, each of which represents a different class. It is noteworthy that there are significant differences in the sample distribution amongst classes, with some having a high sample count and others having a very low sample count.

2.3 Models

In this project, we employed two key models: Normal GAN (Generative Adversarial Network) and WGAN (Wasserstein Generative Adversarial Network). These models are pivotal in the generation and enhancement of a large dataset of Telugu characters, crucial for improving the preservation and usability of historical manuscripts.

GAN (Generative Adversarial Network): GAN architecture consisting of two neural networks: the generator and the discriminator. The generator learns to produce synthetic Telugu character images from random noise, aiming to fool the discriminator into believing these images are real. Meanwhile, the discriminator is trained to distinguish between real Telugu character images from the dataset and fake images generated by the generator. Through adversarial training,GAN iteratively improves both the generator's ability to create realistic Telugu characters and the discriminator's ability to differentiate between real and generated images.

WGAN (Wasserstein Generative Adversarial Network): WGAN is an advanced variant of GAN that introduces the Wasserstein distance metric for training stability. Unlike traditional GANs, WGAN focuses on optimizing the discriminator to approximate the

Wasserstein distance between the distributions of real and generated samples. This approach leads to more stable training dynamics and encourages the generator to produce higher-quality Telugu character images. WGAN has shown benefits in mitigating mode collapse and improving the overall quality of generated samples compared to GAN.

Chapter 3

Code

3.1 Code With Outputs

Import necessary libraries

```
import os

import numpy as np

from PIL import Image

import tensorflow as tf

from tensorflow.keras import layers

import matplotlib.pyplot as plt

import time

from IPython import display
```

Visualizing the dataset: This Python script loads grayscale PNG images from a specified directory using PIL. It checks if any images were loaded and displays up to 25 in a 5x5 grid using Matplotlib. If no PNG images are found, it prints a message. This script efficiently handles loading and visualization of images for exploration or further processing.

```
def load_custom_dataset(dataset_path):

    images = []
    for root, _, files in os.walk(dataset_path):
        for filename in files:
            if filename.endswith('.png'):
                image_path = os.path.join(root, filename)
                image = Image.open(image_path).convert('L')
                images.append(image)
    return images

dataset_path = '/content/telugu_dataset'
images = load_custom_dataset(dataset_path)

if images:
    plt.figure(figsize=(10, 10))
    num_images_to_plot = min(25, len(images))
    for i in range(num_images_to_plot):
        plt.subplot(5, 5, i + 1)
        plt.imshow(images[i], cmap='gray')
        plt.axis('off')
        plt.title(f'Image {i+1}')
    plt.tight_layout()
    plt.show()
else:
    print("No PNG images found in the specified directory.")
```

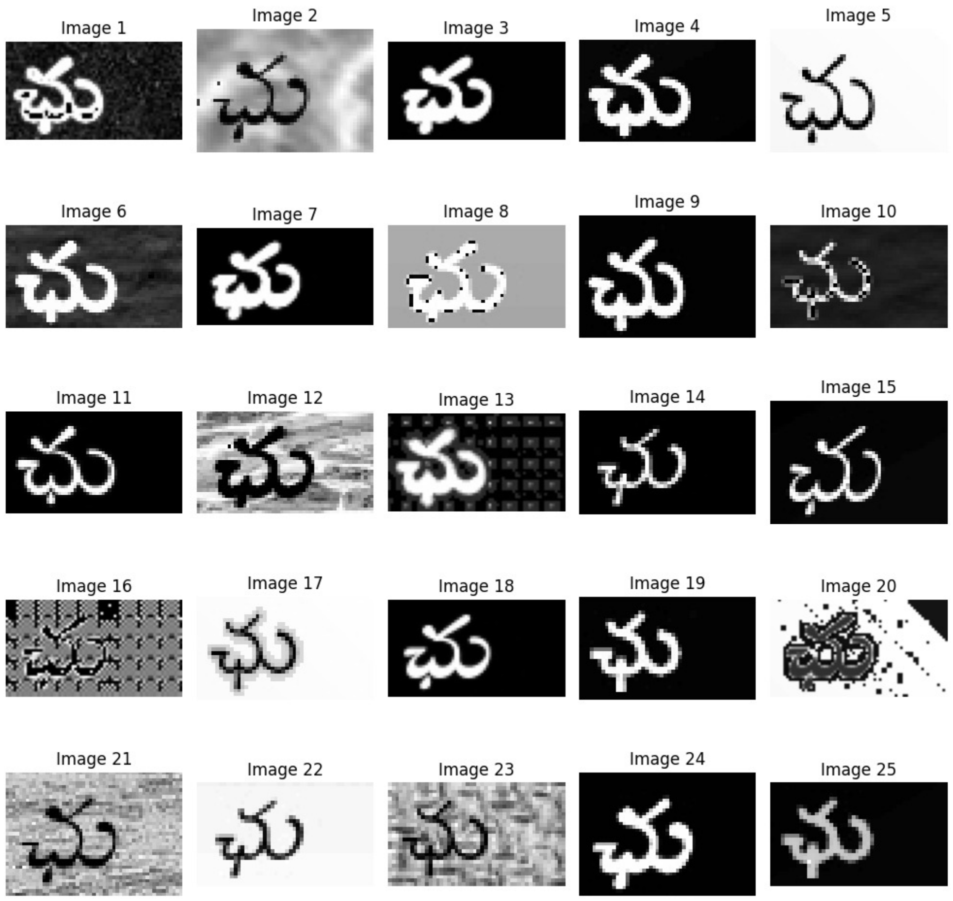

Figure 3.1: input image

WGAN Architecture:

Generator Model: This code defines a generator model for a Generative Adversarial Network (GAN) using TensorFlow/Keras. It starts with a dense layer transforming a 100-dimensional random noise vector into a larger tensor. Batch normalization and LeakyReLU activation normalize the tensor, which is then reshaped into a 3D format and upsampled through Conv2DTranspose layers. The final layer uses a tanh activation function to ensure pixel values within the range [-1, 1]. The model is crucial for image synthesis and enhancement tasks.

```
def make_generator_model():

    model = tf.keras.Sequential()

    model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256)

    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
        padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
        padding='same', use_bias=False))
    assert model.output_shape == (None, 14, 14, 64)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same',
        use_bias=False, activation='tanh'))
    assert model.output_shape == (None, 28, 28, 1)

    return model

generator = make_generator_model()
noise = tf.random.normal([1, 100])
generated_image = generator(noise, training=False)
plt.imshow(generated_image[0, :, :, 0], cmap='gray')
plt.show()
```

Critic Model: This code defines a critic model using TensorFlow/Keras to evaluate images and distinguish between real and generated ones. It uses two Conv2D layers with LeakyReLU activation and dropout regularization to extract and process image features. The flattened output is passed through a Dense layer, which outputs a scalar value representing the critic's decision. The model evaluates a generated image to produce a decision score, aiding in GAN training by guiding the generator towards more realistic outputs.

```
    def make_critic_model():
    model = tf.keras.Sequential()
    model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
        input_shape=[28, 28, 1])) # 2x2 downsample
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))

    return model

critic = make_critic_model()
decision = critic(generated_image)
print(decision)
```

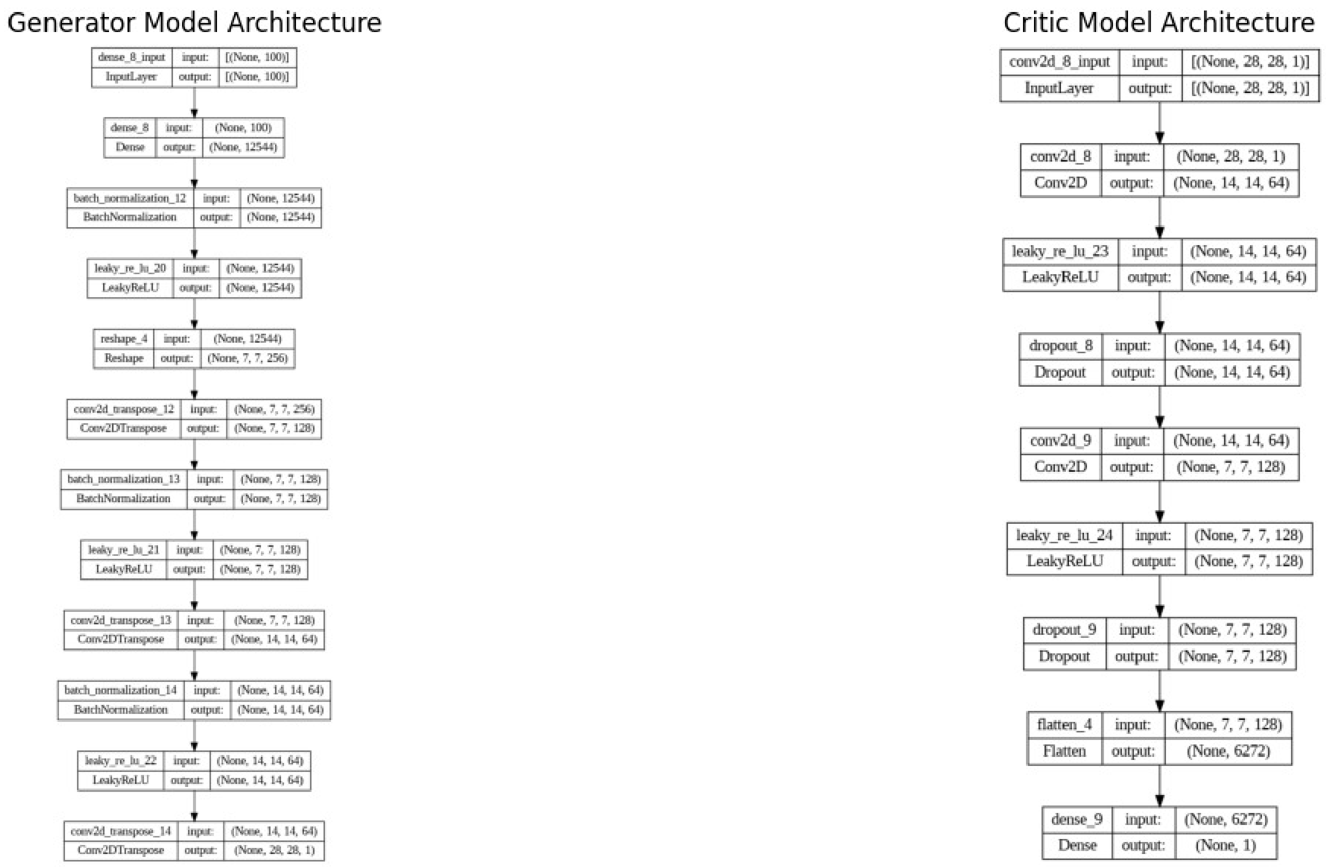
Generator Model Architecture
dense_8_input | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
dense_8 | input: | (None, 100)
Dense | output: | (None, 12544)
batch_normalization_12 | input: | (None, 12544)
BatchNormalization | output: | (None, 12544)
leaky_re_lu_20 | input: | (None, 12544)
LeakyReLU | output: | (None, 12544)
reshape_4 | input: | (None, 12544)
Reshape | output: | (None, 7, 7, 256)
conv2d_transpose_12 | input: | (None, 7, 7, 256)
Conv2DTranspose | output: | (None, 7, 7, 128)
batch_normalization_13 | input: | (None, 7, 7, 128)
BatchNormalization | output: | (None, 7, 7, 128)
leaky_re_lu_21 | input: | (None, 7, 7, 128)
LeakyReLU | output: | (None, 7, 7, 128)
conv2d_transpose_13 | input: | (None, 7, 7, 128)
Conv2DTranspose | output: | (None, 14, 14, 64)
batch_normalization_14 | input: | (None, 14, 14, 64)
BatchNormalization | output: | (None, 14, 14, 64)
leaky_re_lu_22 | input: | (None, 14, 14, 64)
LeakyReLU | output: | (None, 14, 14, 64)
conv2d_transpose_14 | input: | (None, 14, 14, 64)
Conv2DTranspose | output: | (None, 28, 28, 1)
Critic Model Architecture
conv2d_8_input | input: | [(None, 28, 28, 1)]
InputLayer | output: | [(None, 28, 28, 1)]
conv2d_8 | input: | (None, 28, 28, 1)
Conv2D | output: | (None, 14, 14, 64)
leaky_re_lu_23 | input: | (None, 14, 14, 64)
LeakyReLU | output: | (None, 14, 14, 64)
dropout_8 | input: | (None, 14, 14, 64)
Dropout | output: | (None, 14, 14, 64)
conv2d_9 | input: | (None, 14, 14, 64)
Conv2D | output: | (None, 7, 7, 128)
leaky_re_lu_24 | input: | (None, 7, 7, 128)
LeakyReLU | output: | (None, 7, 7, 128)
dropout_9 | input: | (None, 7, 7, 128)
Dropout | output: | (None, 7, 7, 128)
flatten_4 | input: | (None, 7, 7, 128)
Flatten | output: | (None, 6272)
dense_9 | input: | (None, 6272)
Dense | output: | (None, 1)

Figure 3.2: WGAN Architecture

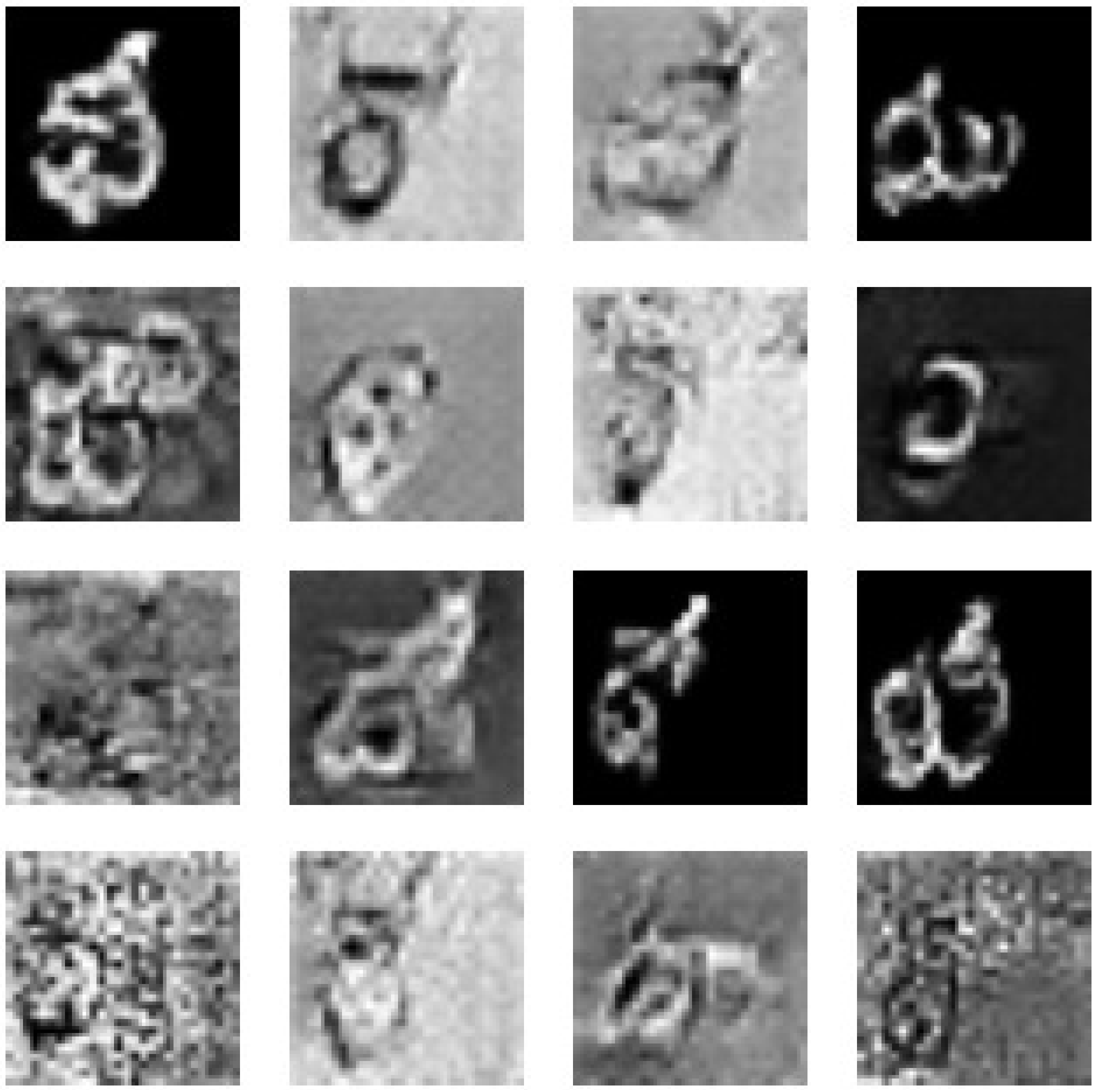

Figure 3.3: The output image generated after 1000 epochs

Chapter 4

Conclusion and Future Work

In summary, this study uses Generative Adversarial Networks (GANs) to create a large dataset of Telugu characters, thereby addressing the fundamental problem of developing efficient machine learning models for Telugu language processing. One major obstacle in the field that has impeded progress in Telugu character identification and cultural preservation efforts is the lack of large, diverse datasets.

We have successfully enlarged our dataset to 38,915 images across 397 labels, each with a different sample size, by implementing Normal GAN and WGAN models. The richness and complexity of Telugu characters are captured in this dataset, which is saved in the.pkl format and is crucial for building strong machine learning models.

Future Directions:

Moving forward, our study seeks to improve the effectiveness of Telugu language processing models through a number of key breakthroughs. One major focus will be on integrating advanced neural network architectures such as DenseNet, ResNet, and UNet. These architectures are well-known for their capacity to capture subtle elements and relationships in images, which could improve the accuracy and resilience of Telugu character recognition systems.

Furthermore, we intend to further our research into Generative Adversarial Networks (GANs) by experimenting with a wider range of designs and configurations. By using many GANs at the same time, we want to increase the diversity and quality of generated Telugu character images, enriching our dataset and enhancing our machine learning models. These advancements are expected to significantly improve the capabilities of our Telugu language processing framework, fostering advances in optical character recognition (OCR), natural language processing (NLP), and cultural preservation through improved digitization of historical Telugu manuscripts and literature. By pushing the boundaries of AI technology in linguistic diversity and cultural heritage preservation, we hope to make significant contributions to academic research and practical applications in the field.