WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

GAN-Based Handwritten Digit Synthesis: Leveraging Kannada MNIST Database

Spurthi S[1] and Prerna Mahajan[2]

JAIN (Deemed-to-be) University

ABSTRACT.

This paper introduces a Generative Adversarial Network (GAN) for synthesizing realistic handwritten digits using the Kannada MNIST dataset. The GAN employs a generator to create digit images from random noise and a discriminator to distinguish between real and generated samples. Through an adversarial training process, the model achieves a balance, resulting in the generation of authentic-looking handwritten digits. The architecture is designed with convolutional and densely connected layers, and training progress is monitored using adversarial loss. The proposed GAN offers a foundation for further optimization and potential applications in data augmentation and synthetic data generation.

1. Introduction

The generator and discriminator that make up the GAN are trained against each other to produce high-quality digit pictures. Preprocessed and normalized, the Kannada MNIST dataset is divided into training and testing sets. The discriminator separates genuine and created samples, while the generator network converts random noise into artificial digit pictures.

To produce meaningful digit representations, the generator is built with densely connected layers, reshaping, and upsampling procedures. A convolutional neural network that can distinguish between generated and actual numbers is the discriminator. An adversarial process is used to train the GAN, with the generator trying to fool the discriminator and the discriminator trying to accurately categorize generated and real data. In the training phase, both the generator and the discriminator are simultaneously optimized.

The GAN's performance is monitored through the adversarial loss, providing insights into the quality of generated digits. After training for a specified number of epochs, the generator can produce synthetic handwritten digits. The model's efficacy is demonstrated through the generation and display of new digit images.

While this GAN architecture provides a foundation for digit generation, further enhancements and fine-tuning may be necessary for optimal results. Hyperparameter tuning, architectural modifications, and extended training durations could contribute to improved digit generation capabilities. Additionally, proper evaluation metrics and visualizations should be employed to assess the model's effectiveness and convergence. This project lays the groundwork for leveraging GANs in generating handwritten digits with potential applications in data augmentation and synthetic data generation for machine learning tasks.

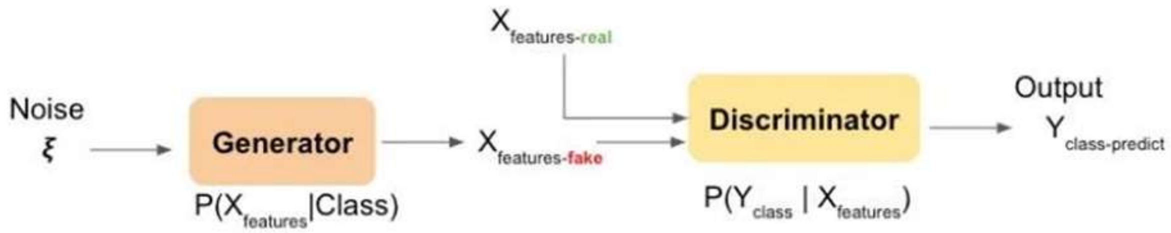

Fig. 1. Simple diagram of GANs model

2. Literature Review

In a variety of disciplines, Generative Adversarial Networks (GANs) have become increasingly popular for producing realistic data samples. This study of the literature explores the state of the art in GAN research, particularly as it relates to the creation of Kannada MNIST digit pictures. The Kannada

MNIST dataset, which consists of handwritten numbers in the Kannada script, has special difficulties that call for customized methods for efficient creation.

Since their introduction by Goodfellow et al. in 2014, Generative Adversarial Networks (GANs) have shown to be an effective technique for producing realistic data samples. They are made up of two neural networks playing a min-max game: a discriminator and a generator. The discriminator works to separate authentic samples from fakes, while the generator creates data samples from noise.

GANs are widely used in many different fields, including medical image synthesis, style transfer, and natural picture production. GANs have shown impressive results in handwritten digit generation, producing realistic digit images for a variety of scripts, such as Devanagari, Latin, and Chinese.

An expansion of the original MNIST dataset, the Kannada MNIST dataset consists of handwritten digit pictures in the Kannada script. It acts as a standard for assessing machine learning algorithms in the Kannada language, with 60,000 training images and 10,000 test images.

Several approaches have been proposed for generating Kannada MNIST digits using GANs. These approaches often involve modifications to network architectures and training strategies to suit the characteristics of the Kannada script. Some methods employ conditional GANs to control the generation process based on class labels, while others explore novel architectures like Wasserstein GANs or progressive growing GANs to enhance stability and quality.

Evaluation of GANs for Kannada MNIST digit generation involves various metrics including visual inspection, inception score, Frechet Inception Distance (FID), and human evaluation. Visual inspection assesses the quality and diversity of generated images, while quantitative metrics like inception score and FID measure the similarity between generated and real images in terms of visual appearance and feature distributions.

Challenges persist in generating Kannada MNIST digits, such as handling imbalanced datasets, accommodating variations in handwriting styles, and ensuring diversity in generated samples. Future research directions may involve exploring advanced network architectures, incorporating linguistic constraints, and leveraging unsupervised or semi-supervised learning techniques to further enhance the quality and diversity of generated images.

In summary, Generative Adversarial Networks offer a promising approach for generating realistic handwritten digit images in the Kannada script. By addressing existing challenges and leveraging advancements in GAN methodologies, further improvements can be achieved, paving the way for diverse applications in Kannada language processing and digit recognition tasks.

3. Features and Dataset

More than 44 million people in South India speak the regional language of Kannada. In August 2019, the Kannada MNIST dataset was released as a component of the Kannada-MNIST study.

Each image has a height of 28 pixels and a width of 28 pixels, for a total of 784 pixels. Every pixel has a unique value that indicates how light or dark it is; darker tones are indicated by greater values. The integer values of the pixels span from 0 to 255, inclusive.

Each pixel column in the training set is identified by the notation "pixel{x}," where x is an integer that falls between 0 and 783, inclusive. Consider breaking down x into the formula $x = i * 28 + j$, where i and j are numbers ranging from 0 to 27, inclusive, to determine the location of this pixel in the image. As a result, pixel{x} can be found using zero-based indexing at the intersection of row i and column j in a 28 x 28 matrix.

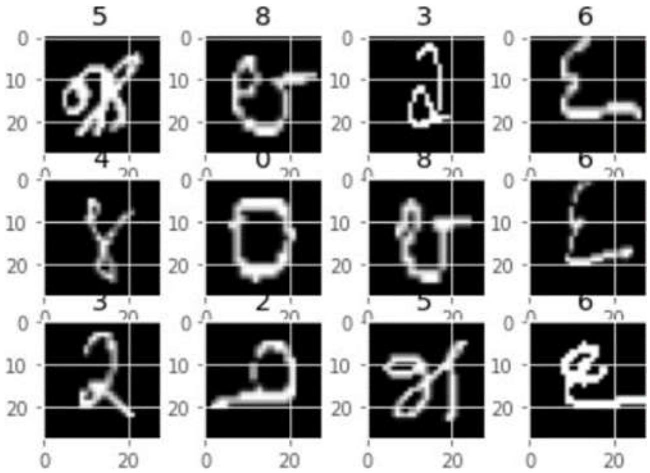

Fig. 2. Sample Kannada digits from Kannada MNSIT

4. Experiments and Findings

4.1 Discriminator

The discriminator simultaneously processes both generated features from the generator and actual features during the learning process. It makes predictions using the binary cross-entropy (BCE) cost function, where the labels fed into the cost function comprise both real and fake features. This approach ensures that the discriminator progressively improves its ability to distinguish between fake and real features over time.

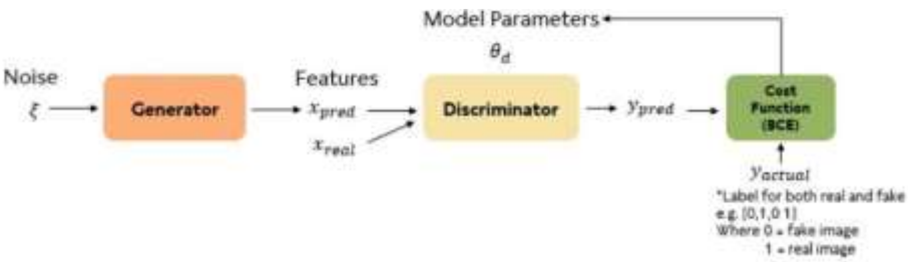

Fig. 3. Training process for discriminator

Model: "Discriminator"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_9 (Conv2D)	(None, 14, 14, 64)	1088
leaky_re_lu_7 (LeakyReLU)	(None, 14, 14, 64)	0
conv2d_10 (Conv2D)	(None, 7, 7, 128)	131200
leaky_re_lu_8 (LeakyReLU)	(None, 7, 7, 128)	0
conv2d_11 (Conv2D)	(None, 4, 4, 256)	524544
leaky_re_lu_9 (LeakyReLU)	(None, 4, 4, 256)	0
flatten_3 (Flatten)	(None, 4096)	0
dense_13 (Dense)	(None, 1)	4097

Total params: 660,929
Trainable params: 660,929
Non-trainable params: 0

Summary of Discriminator Model

4.2 Discriminator

The training process for the generator differs slightly from that of the discriminator. When the generator produces fake features for the discriminator, the discriminator receives only these fake features for prediction. The labels fed into the cost function during this process are marked as all real images. This implies that if the discriminator correctly predicts the fake features as real (contrary to the false label), the generator updates its model parameters in that direction.

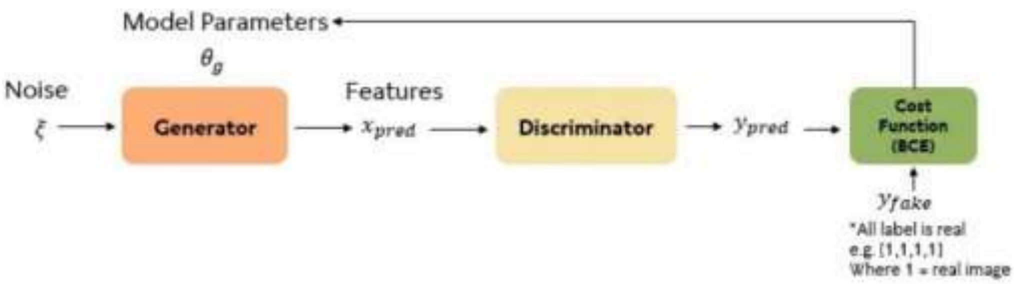

Fig. 4. Training process for Generator

```
Model: "Generator"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 128)]             0
_________________________________________________________________
dense_12 (Dense)             (None, 16384)             2113536
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 16384)             0
_________________________________________________________________
reshape_2 (Reshape)          (None, 8, 8, 256)         0
_________________________________________________________________
up_sampling2d (UpSampling2D) (None, 16, 16, 256)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 16, 16, 128)       524288
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 16, 16, 128)       0
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 32, 32, 128)       262144
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 32, 32, 128)       0
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 64, 64, 3)         6144
=================================================================
Total params: 2,906,112
Trainable params: 2,906,112
Non-trainable params: 0
```

Summary of Generator Model

Conclusion

In conclusion, this research introduces a Generative Adversarial Network (GAN) specifically designed for synthesizing realistic handwritten digits using the Kannada MNIST dataset. The GAN's adversarial training process, involving a generator and discriminator, results in the generation of authentic-looking digits. The architecture, utilizing densely connected layers and convolutional neural networks, demonstrates efficacy in transforming random noise into meaningful digit representations. The paper highlights the potential applications of the GAN in data augmentation and synthetic data generation for machine learning tasks. While acknowledging the presented foundation, further optimization through hyperparameter tuning, architectural modifications, and extended training durations is suggested for enhanced digit generation capabilities. The importance of employing proper evaluation metrics is emphasized, positioning this research as a stepping stone for future advancements in the field.

Reference

[1] Prabhu, V. U. (2019). Kannada-MNIST: A new handwritten digits dataset for the Kannada language. *arXiv preprint arXiv:1908.01242*.

[2] Shrisha, H. S., Anupama, V., Suresha, D., & Jagadisha, N. (2022). KGAN: A Generative Adversarial Network Augmented Convolution Neural Network Model for Recognizing Kannada Language Digits. In *Communication and Intelligent Systems: Proceedings of ICCIS 2021* (pp. 523-531). Singapore: Springer Nature Singapore.

[3] Mahapatra, D., Choudhury, C., & Karsh, R. K. (2020, December). Generator based methods for off-line handwritten character recognition. In *2020 Advanced Communication Technologies and Signal Processing (ACTS)* (pp. 1-6). IEEE.

[4] Ganesh, A., Jadhav, A. R., & Pragadeesh, K. C. (2018). Deep learning approach for recognition of handwritten kannada numerals. In *Proceedings of the Eighth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2016)* (pp. 294-303). Springer International Publishing.