Using Generative Adversarial Networks in Classification Tasks with Very Small Amounts of Training Data

GOFUR HALMURATOV, ARNOŠT VESELÝ
Department of Information Engineering,
Czech University of Life Sciences,
Kamýcká 129, 165 00 Praha - Suchdol,
CZECH REPUBLIC

Abstract: - Deep learning algorithms are incredibly powerful and have achieved impressive results in various classification tasks. However, one of their limitations is their dependence on large amounts of training data. When there is limited training data available, the standard approach is to increase the dataset size by using data augmentation and training a neural network on the expanded dataset. This method can be effective, but it requires significant computational resources and may not always be feasible. In our work, we proposed a new approach to address the problem of limited training data. Instead of relying solely on increasing the dataset size, we train Generative Adversarial Networks (GANs) to generate distributions of individual categories. The classification of the unknown element is then performed using distributions generated by trained GAN networks. We proposed four methods that compare the unknown element with the elements generated by trained GAN networks and establish estimates of the conditional probabilities of the unknown element belonging to individual categories. These conditional probabilities are then used for the classification of the unknown element into individual categories. This approach enables us to make informed decisions and achieve accurate classification results even when dealing with limited training data.

Key-Words: - Preparing data, Deep learning algorithms, Latent space, Generative adversarial network(GAN), Variational autoencoder(VAE)

Received: June 28, 2022. Revised: May 19, 2023. Accepted: June 23, 2023. Published: July 19, 2023.

1 Introduction

Classification is a fundamental task in machine learning and computer vision, aimed at predicting the class label of input data based on a pre-defined set of categories, [1]. The performance of a classifier heavily relies on the quality and size of the available training data, [2]. However, in many real-world scenarios, the amount of training data is limited, posing challenges such as overfitting or underfitting that can significantly impact the classifier's performance, [3], [4]. To address the limitations of small training datasets, researchers have proposed various techniques, including data augmentation, transfer learning, and ensemble methods, [5], [6], [7]. In recent years, generative adversarial networks (GANs) have gained significant attention due to their ability to generate new data samples that closely resemble the original training data, [8]. By augmenting the training dataset, GANs have shown promise in improving the performance of classifiers when faced with limited training data, [9], [10]. However, traditional approaches often focus on expanding the dataset size, requiring substantial computational resources, and may not always be feasible. In this article, we propose a novel approach to address the problem of limited training data. Instead of solely relying on increasing the dataset size, we leverage trained GAN networks in the decision-making phase. During this phase, the trained GAN networks simulate the conditional probabilities of individual categories, enabling us to estimate the conditional probabilities of unknown elements and make informed classification decisions, [11]. We present two methods based on trained GAN networks, along with their two modifications. The first modification involves using GAN networks trained for individual categories to train a Variational Autoencoder (VAE), [12]. The VAE can generate conditional distributions of all individual categories, and in the decision-making phase, only the variational autoencoder is utilized, [13]. The second modification utilizes GAN networks trained for individual categories to transform the decision process into the latent space of the autoencoder, [14]. By leveraging the learned representations in the latent space, the decision-making process

becomes more efficient and effective, [15]. By employing these methods, we aim to achieve accurate classification results even with limited training data. The proposed approach offers an alternative to traditional data augmentation techniques, reducing the reliance on dataset size expansion and providing a more efficient solution for classification tasks, [16]. In the following sections, we provide a detailed description of the proposed methods and present experimental results to validate their effectiveness. By leveraging the power of GAN networks and incorporating them into the decision-making process, we can overcome the limitations of limited training data and enhance the performance of classifiers in real-world scenarios.

2 Data

The MNIST handwritten digits database was utilized in the training and evaluation of our proposed methods. The MNIST dataset contains 10 distinct categories of handwritten digits. To properly train and evaluate our methods, we shuffled the MNIST database and selected a small subset of the data for training. We conducted experiments with various numbers of training data, including 1, 2, 5, 10, and 100 per category, and evaluated the performance of our proposed methods using 100 testing data samples per category (Figure.1).

Fig. 1: Training data sample

3 Methodology

The basic way of using GANs in a classification task is as follows. Suppose, we classify vectors $x \in X$ into k disjoint classes $C_1, C_2, \dots, C_k$. First, we will create a neural network with GAN architecture. We then train this network k times on the data of individual categories C_i. We thus obtain k trained networks GAN_i that simulate probability distributions of individual categories $P_{C_i}(x)$ (Figure.2). We then use the discriminators of the trained networks GAN_i for classification by inserting the classified vectors $\overline{x}$ into the input of the discriminator and treating the output of the discriminator as a probability estimate $P(\overline{x} \in C_i)$. This is the basic way to proceed.

There is another way. When classifying the vector x, we first approximate individual probability distributions $P_{C_i}(x)$ by generating a certain number of their realizations and then comparing the vector $\overline{x}$ with these approximations. In this work, we have proposed several methods by which this comparison can be made.

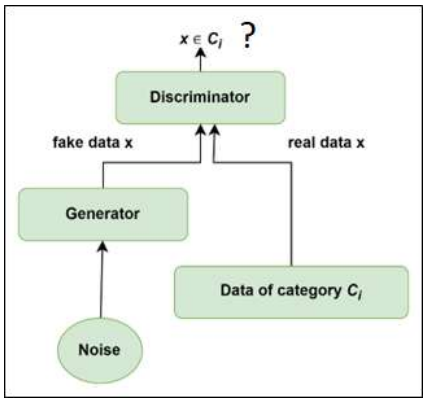

Fig. 2: Training of the network GAN_i

Method 1(M1): Classifies an unknown vector $\overline{x}$ in the following way:

1) For each distribution $P_{C_i}(x)$, $i = 1, \dots, k$ the method
   a. Generates m vectors $x_1, \dots, x_m$ with the help of trained GAN for category C_i.
   b. Determines distance values $d_1 = |\overline{x} - x_1|, \dots, d_m = |\overline{x} - x_m|$
   c. Sorts the values $d_1, \dots, d_m$ in ascending order.
   d. Determines the distance of the pattern $\overline{x}$ from the category C_i as follows: $d(\overline{x}, C_i) = \frac{1}{L}\sum_{i=1}^{L} d_i$, where L is the optional parameter of the method.
   e. Finally classifies the unknown vector $\overline{x}$ into that category j for which the value $d(\overline{x}, C_j)$, $j = 1, \dots, k$ is the smallest.
2) Point 1) is repeated N times. We denote by $N_1, \dots, N_k$ the number of classifications of categories $C_1, \dots, C_k$. $N = N_1 + \cdots + N_k$.

3) Finally, the method estimates each category $C_i,\ i = 1, \dots, k$ its probability: $P(C_i) = \frac{N_i}{N},\ i = 1,\ \dots, k.$

Method 2(M2): This method uses the distributions $P_{C_1}(x),\ \dots,\ P_{C_k}(x)$ to train the variational autoencoder so that it is able, after inserting a vector $\overline{x}$ from any category $C_1,\ \dots, C_k$ into its input to reproduce it on its output. The autoencoder can thus simulate the probability distribution $P_C(x),\ C = C_1 \cup C_2 \cup \dots \cup C_k,\ C_i \cap C_j = \emptyset,\ i, j = 1, \dots, k$

We denote the latent space of the autoencoder by Z. The learned autoencoder first transforms the input vector $\overline{x}$ to $\overline{z} \in Z$ and then transforms $\overline{z}$ into the output vector $\overline{y} \triangleq \overline{x}$ (see Figure.3).

Method 2 uses method 1 above for classification, but does it in the latent space Z of the autoencoder:

1) For each distribution $P_{C_i}(x)$:
   a. Generates m vectors $x_1,\ \dots,\ x_m$ with the help of trained GAN for category C_i.
   b. Determines distance values $d_1 = |\overline{z} - z_1|,\ \dots,\ d_m = |\overline{z} - z_m|$, where $\overline{z}$ is the projection of the vector $\overline{x}$ into the latent space Z.
   c. Sorts value $d_1,\ \dots,\ d_m$ in ascending order.
   d. Determines the distance of the vector $\overline{x}$ from the category $C_i$as follows: $d(\overline{x}, C_i) = \frac{1}{L}\sum_{i=1}^{L} d_i$, where L is the optional parameter of the method.
   e. Finally classifies the unknown vector $\overline{x}$ into the category j for which the value $d(\overline{x}, C_j),\ j = 1,\ \dots,\ k$ is the smallest.

Points 2) and 3) of Method 2 are the same as for Method 1.

Method 3(M3): This method is a modification of method 1 (M1):

The method estimates the probabilities $P(x \in \varepsilon(\overline{x}) \mid x \in C_i),\ i = 1, \dots, k,$ where $\varepsilon(\overline{x})$ is the ε neighborhood of the vector $\overline{x}$, using simulation of distributions, $P_{C_1}(x),\ \dots,\ P_{C_k}(x),\ x \in X$. The neighborhood $\varepsilon(\overline{x})$ is defined as:

$$x \in \varepsilon(\overline{x}) <=> |x - \overline{x}| < \varepsilon.$$

Suppose we generate for each category m vectors. Let the n_i vectors from category C_i, $i = 1,\ \dots,\ k$ fall into $\varepsilon(x)$. We can then estimate the probabilities $P(x \in \varepsilon(\overline{x}) \mid x \in C_i),\ i = 1, \dots, k$ as follows:

$$P(x \in \varepsilon(\overline{x}) \mid x \in C_i) = \frac{n_i}{m},\ n = \sum_{i=1}^{k} n_i$$

From this and Bayes's formula, it follows:

$$P\big(x \in C_i \mid x \in \varepsilon(\overline{x})\big) = \frac{P(x \in \varepsilon(\overline{x}) \mid x \in C_i)\, P(x \in C_i)}{P\big(x \in \varepsilon(\overline{x})\big)}$$

Since we generated the same number of elements from each category, $P(x \in C_i) = \frac{1}{k}$ and since n vectors fall into the $\varepsilon(\overline{x})$ we set $\big(x \in \varepsilon(\overline{x})\big) = \frac{n}{m \cdot k}$. The question arises of how to choose the size of the $\varepsilon(\overline{x})$ so that the appropriate number of generated vectors falls into it. For example, we want that $\mu = \frac{n}{m \cdot k}$ be 0.1.

We can proceed as follows:

1) For all categories, we generate m vectors x_i.
2) We determine the distances of generated vectors x_i from $\overline{x}$, $d_i = |\overline{x} - x_i|$, and we sort them in ascending order.
3) Then we put $\varepsilon = |\overline{x} - x_n|$, where $n = \mu \cdot m \cdot k$

Method 4(M4): We can modify Method 2 in the same way as Method 3. This will be the method 4(M4).

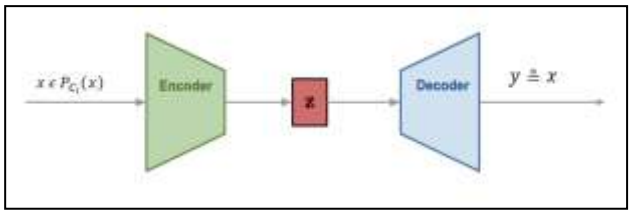

Fig. 3: Variational autoencoder that learns $P_{C_i}(x)$

4 Results

We experimented with four proposed classification methods M1, M2, M3, and M4. They were evaluated using a small number of training data, specifically 1, 2, 5, 10, and 100 instances per category from the MNIST handwritten digits database. The performance of the proposed methods was measured using a testing dataset consisting of 100 instances per category, totaling 1000 instances. The Generative Adversarial Network (GAN) is trained on the MNIST dataset, which is filtered to only include one digit category, resulting in 10 separate GANs (GAN_i). Each GAN consists of two main components: a generator and a discriminator. The generator model has a 100-dimensional latent input layer and is composed of three dense layers with batch normalization and ReLU activation. On the other hand, the discriminator model takes an image of 28x28 pixels as input and is constructed of three dense layers with ReLU activation, as well as an output layer with a sigmoid activation (see Model 3.). All models were optimized using binary cross-entropy loss and Adam optimizer with a learning rate of 0.0002 and a decay rate of 0.5. The training of each GAN_i per digit, the category was carried out for 30,000 epochs. Due to the limited size of the training data, the process was efficient and took around 2-3 hours for each GAN model. GAN_i generators were trained for individual data variants of the following 1, 2, 5, 10, and 100 instances per category. After they have been trained, they can be used for classification, because the output of individual GAN_i we have probability estimates of individual $P(x \epsilon C_i)$ categories. The results of the proposed methods M1, M2, M3, and M4 were compared with the discriminator results of the GAN model, which was used as a baseline for comparison. The discriminator's classification accuracy was 38.2% for 1 training instance per category, 42.1% for 2 training instances per category, 56.9% for 5 training instances per category, 61.2% for 10 training instances per category, and 89.3% for 100 training instances per category. These results are summarized in column 'Discriminator result accuracy' in Table 1. In the initial stage of our experiments, we evaluated the performance of Method 1 with various parameter combinations. To ensure compatibility with low-performance computers, we selected the parameters m=100 and N=1 as a reasonable configuration for conducting experiments. For the M3, we used parameters m=100 and $= 0.1$. The results of classification using M1 and M3 were evaluated using different amounts of training data from each category of the MNIST handwriting digit dataset. The classification accuracy of M1 ranged from 41.3% to 66.4% and that of M3 ranged from 41.9% to 66.7%. The change ratio in percentage compared to the discriminator's classification result showed that M1 performed better than the discriminator with 1-2 training data from each category, but the performance worsened as the amount of training data increased. The same trend was observed for M3, with a slightly better improvement over the discriminator for all amounts of training data compared to M1. Overall, the results indicate that both M1 and M3 performed better than the discriminator for small amounts of training data, but as the amount of data increased, their performance degraded compared to the discriminator's classification result. After experiments with methods M1 and M3, we carried out experiments with M2 and M4. The Variational autoencoder(VAE) of these methods uses the MNIST dataset, which contains images of handwritten digits, as input. The encoder network maps the input data to a lower-dimensional space (latent space=32), and the decoder network maps back from the latent space to the original input data. The sampling function implements the reparameterization trick, which is a technique to ensure that the sampling of the latent code z is differentiable and can be backpropagated during training. The VAE encoder network consists of several Conv2D, Flatten, and Dense layers(see Model 1.), and the VAE decoder network consists of several Conv2DTranspose, Reshape, and Dense layers(See Model 2.).The VAE is trained using the mean squared error loss. In the initial stage of our experiments, we evaluated the performance of Method 2 with various parameter combinations. To ensure compatibility with low-performance computers, we selected the parameters m=100 and N=1 as a reasonable configuration for conducting experiments. For the modification of M2 (Method 4), we used parameters m=100 and $\mu = 0.1$ See Table 1. for more details. The proposed methods showed better results with smaller training data sets, but their performance worsened as the number of training data increased. The modified versions of

Method 1 (M3) and Method 2 (M4) performed better than their original forms (M1 and M2) respectively. The best results were obtained with the smallest training data sets (1, 2, 5, and 10). However, the proposed methods performed poorly with larger training data sets, with the best performance being 16,75% better and the worst being 21.16% worse.

5 Discussion

In our article, we propose a new way to use GAN networks for solving classification tasks for which we only have a limited amount of training data. The learned GAN networks are not used to expand the training set and to subsequently learn a layered convolutional network. Instead, we will use the learned GAN networks only in the decision-making phase to simulate the conditional distributions of individual categories. Although this simulation is computationally somewhat more difficult than obtaining the decision of a trained convolutional network, it is still manageable. The results we obtained show that algorithms of this type could give better results for very small training sets. For larger data sets, it turns out that the classic method of expanding the training set and then learning a classic convolutional neural network on the expanded training set will probably give better results. Using a more complex architecture of GAN and VAE networks and further experimenting with parameters during their learning would probably lead to an increase in classification accuracy. The results of our experiments indicate that the data generated by the trained GAN has a significant amount of noise and deviates from the original data. To overcome this limitation, we propose to investigate and implement noise removal techniques in the GAN-generated training data in future work. Our evaluation of the testing data showed that the proposed classification methods are ineffective when applied to rotated data. To resolve this, we recommend augmenting the training data with rotations. By adding rotated versions of the training data, the classification models would have more exposure to this type of variation and may be better equipped to handle it. Additionally, it may be useful to explore other data augmentation techniques beyond rotations, such as scaling, flipping, or adding noise, to further improve the robustness of the models to different types of variations. Overall, it is important to carefully experiment and evaluate different approaches to data augmentation to find the best strategy for improving the performance of the proposed classification methods.

6 Conclusion

It's good that the proposed classification methods hold the potential for improving performance when limited training data is available. Even if they are not significantly better than the existing GAN discriminator model, they may still have value in certain contexts. Regarding the suggestion to experiment with more complex models and training parameters, it's important to carefully consider the trade-offs between model complexity and generalization performance. Increasing the complexity of the models and the number of training parameters can lead to better performance on the training data, but can also increase the risk of overfitting and decrease performance on new, unseen data. Therefore, it's important to evaluate the performance of the models on both the training and testing data and use techniques such as cross-validation to ensure that the models are not overfitting. Additionally, it may be helpful to explore other approaches to improving performance, such as ensemble methods or transfer learning, which can leverage the strengths of multiple models and improve generalization performance without requiring significant increases in model complexity.

7 Future Work

To further improve the performance of classification methods M2 and M4 we propose to combine the learning of the autoencoder. VAE with clustering. Combining clustering of the latent space of the autoencoder is an interesting idea that could potentially improve the performance of the proposed classification methods. By clustering the latent space, it may be possible to identify more meaningful and interpretable subgroups of data that can be used to improve the accuracy of the classification models. However, it's important to note that clustering in the latent space can also be challenging, as it requires selecting appropriate clustering algorithms, hyperparameters, and evaluation metrics. Additionally, it may be necessary to balance the trade-off between increasing the complexity of the model and improving its performance, as clustering can add additional computational costs and may also increase the risk of overfitting. Overall, the proposed idea of combining clustering with classification methods is promising, but it will require careful experimentation and evaluation to determine its effectiveness. Lastly, it is recommended to conduct experiments using higher computational resources to obtain more accurate and

reliable results. With more computational resources, it may be possible to train larger and more complex models, increase the number of training epochs, and experiment with a wider range of hyperparameters, which could lead to better performance and more robust conclusions. It should be noted that the experiments were conducted using personal computers with limited computational resources, resulting in small training models and a limited number of training epochs for both GANs and VAEs. We recommend repeating the experiments using higher computational resources for better results.

References:

[1] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521 (7553), p.436-444.

[2] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530.

[3] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15 (1), p.1929-1958.

[4] Caruana, R. (2017). Multitask learning. arXiv preprint arXiv:1706.05098.

[5] Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. (2019). AutoAugment: Learning augmentation strategies from data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 113-123.

[6] Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Advances in neural information processing systems, pp. 3320-3328.

[7] Dietterich, T. G. (2000). Ensemble methods in machine learning. In Multiple classifier systems, pp. 1-15.

[8] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems, pp. 2672-2680.

[9] Antoniou, A., Storkey, A., & Edwards, H. (2017). Data augmentation generative adversarial networks. arXiv preprint arXiv:1711.04340.

[10] Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2019). Analyzing and improving the image quality of StyleGAN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8110-8119.

[11] Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., & Metaxas, D. N. (2018). StackGAN++: Realistic image synthesis with stacked generative adversarial networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41 (8), p.1947-1962.

[12] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114.

[13] Chen, T. Q., Li, X., Grosse, R., & Duvenaud, D. K. (2018). Isolating sources of disentanglement in variational autoencoders. In Advances in Neural Information Processing Systems, pp. 2610-2620.

[14] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

[15] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313 (5786), p.504-507.

[16] Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621.

Appendix

Model 1. VAE encoder model

Layer(type)	Output Shape	Param#	Connected to
encoder_input(InputLayer)	[(None,28,28,1)]	0	[]
conv2d(Conv2D)	(None,14,14,512)	5120	['encoder_input[0][0]']
conv2d_1(Conv2D)	(None,7,7,1024)	4719616	['conv2d[0][0]']
flatten(Flatten)	**(None,50176)**	0	['conv2d_1[0][0]']
dense(Dense)	(None,400)	20070800	['flatten[0][0]']
z_mean(Dense)	(None,100)	40100	['dense[0][0]']
z_log_var(Dense)	(None,100)	40100	['dense[0][0]']
z(Lambda)	(None,100)	0	['z_mean[0][0]','z_log_var[0][0]']

Model 2. VAE decoder model

Layer (type)	Output Shape	Param #
z_sampling(InputLayer)	[(None,100)]	0
dense_1(Dense)	(None,50176)	5067776
reshape(Reshape)	(None,7,7,1024)	0
conv2d_transpose(Conv2DTranspose)	(None,14,14,1024)	9438208
conv2d_transpose_1(Conv2DTranspose)l	(None,28,28,512)	4719104

Model 3. Discriminator model of the GAN

Layer (type)	Output Shape	Param #
Input layer (InputLayer)	[(None, 784)]	0
Dense_1 (Dense)	(None, 512)	401920
Dense_2 (Dense)	(None, 256)	131328
Dense_3 (Dense)	(None, 128)	32896
Output_4(Dense)	(None, 1)	129

Table 1. Results of four proposed methods.

	Number of train data	Number of test data	Discriminator result Accuracy	Proposed Method Result accuracy	Ratio in percentage
M1	1	100	38,2	41,3	108,12%
	2	100	42,1	43,6	103,56%
	5	100	56,9	57,4	100,88%
	10	100	61,2	61,8	100,98%
	100	100	89,3	66,4	74,36%
M3	1	100	38,2	41,9	109,69%
	2	100	42,1	43,8	104,04%
	5	100	56,9	58,3	102,46%
	10	100	61,2	61,3	100,16%
	100	100	89,3	66,7	74,69%
M2	1	100	38,2	44,6	116,75%
	2	100	42,1	47,1	111,88%
	5	100	56,9	63,2	111,07%
	10	100	61,2	67,4	110,13%
	100	100	89,3	71,3	79,84%
M4	1	100	38,2	44,2	115,71%
	2	100	42,1	47,3	112,35%
	5	100	56,9	62,9	110,54%
	10	100	61,2	67,2	109,80%
	100	100	89,3	71,9	80,52%

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

-Ing. Gofur Halmuratov is the main author of the research, responsible for the formulation of the problem, development and implementing of the proposed methods, conducting experiments, analyzing results, and writing the article.
-Doc. Ing. Arnošt Veselý, CSc. contributed to the research by providing guidance and support as the advisor, particularly in the development of the proposed methods.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors declare no conflicts of interest.