Recognition of handwritten digits and letters by Image Classification

Project Report for the Course on Machine Learning and Deep Learning (650.025, 23W), University of Klagenfurt

Stepan Fatov (12306851),(fat0010)
stfatov@edu.aau.at, stepan.fatov.st@vsb.cz
14.01.2024, Klagenfurt, Austria

***Abstract*—Human vision is an incredible phenomenon. It's not even a problem for humans to recognize digits or letters. Automating collecting and processing the information is necessary if the information is increasing. Humans' abilities are limited. Classical programming is also very limited in the automation process. Machine learning (ML) and Deep Learning(DL) can help us to process the information. However, there isn't a classical way to exactly design the model to solve the problem. To solve issues with image classification convolution neural networks (CNN) can be very useful. It's also interesting how well classical ML is suitable to solve issues with image classification. Nevertheless, image classification CNNs can be a base for object detection models. And connecting different models such as models for image segmentation, natural language processing, image classification and so on we can build real artificial intelligence (AI). The first that humans start to do when are born is to see. After some time we start to learn how to recognize digits and letters. Because of that, it is interesting to build our neural architecture which will have almost the same accuracy as humans to recognize.**

***Index Terms*—Machine learning, deep learning, image classification, convolution neural networks**

I. Motivation

Rising computing power gives us the ability to build more complex neural networks. That means that we become able to build neural networks that can even recognize digits and letters with the same performance as humans. It allows us to automate a lot of different processes by extracting information from papers or even reading information on the road about dangerous situations. It can be pretty useful for banking systems or for building autonomous vehicles, respectively. Hospitals also have the deal with a lot of papers. Humanity is increasing and it means that the number of consumer banking systems and the number of potential patients are increasing too. The current law system requires using analog format as primary. However, using only papers we are very limited in increasing the whole system. Without using computers and digital technology such as AI we are losing a lot of time to do routine. This moment makes doing research and producing innovation impossible. We are very limited in time. We need more time to do research and become smarter to improve the already-existing world.

Dealing with handwritten digits or letters becomes impossible for classical way programming. Every person has own handwritten style which is developed during growing up. Even now to confirm the identity of the person we are using a signature. There are a lot of ways for example digit 1 can be written.

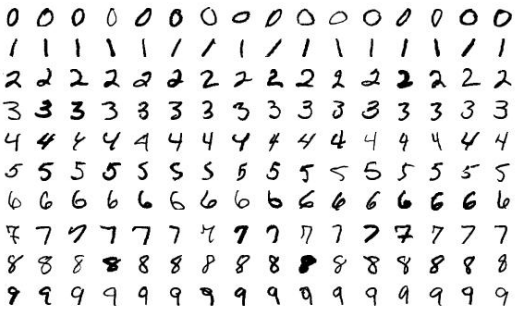

Fig. 1: Different handwritten digits

When we are looking and trying to recognize digits or letters, we don't even think how we are doing it. We are doing it by shape or we are comparing already given patterns in the brain with what we are seeing. It is a simple task to do, but it is very hard to say how it is being done. Research in biology shows us that the human brain consists of small parts. It is called a neuron. Neuron receives an electrical signal, process signal, and sends output to the next neurons. Humans learn from the data which are around us. As we grow up as more we know and make better decisions (better signal processing). In 1957 Frank Rosenblatt published the first perceptron rule. With this rule, he also proposed an algorithm to learn machines from data [1].

II. Data

To learn the model is necessary to have a good-quality data. Figure 2 shows one sample of each class from the dataset. Each image has the same size 28x28 pixels in grayscale. In the base of the dataset lies two sets from the NIST database, MNIST and A-Z letters. They perfectly match each other. Another good point is that it contains a huge amount of different handwritten styles.

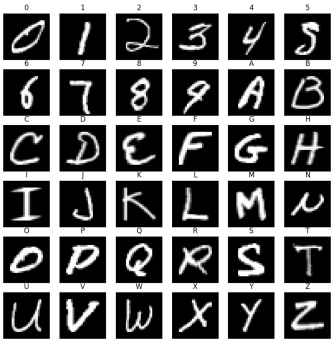

Fig. 2: Example of dataset

To keep the balance between classes for each class, only 2000 samples. There is one issue for class F and I. The whole dataset contains less that 2000 samples for these two classes.

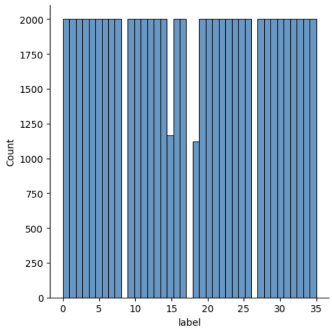

Fig. 3: Number of samples for each class

As we can see there is a big collision between two classes, the digit 0 and the letter O. Even for humans, we need additional information to determine if it is a digit or a letter. One of the way to solve this problem is to take another CNN (sub CNN) which will perfectly know how to determine O or 0.

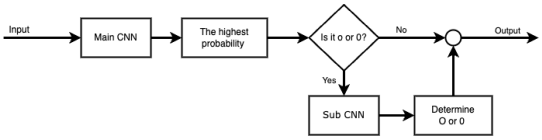

Fig. 4: Solve a collision between 0 and o

Working with RAW images is a complicated task. The reason is that RAW image consists of integer values in the range from 0 to 255. Calculating gradient descent for integer data type is impossible mission. Because of that data has to be normalized. There are three types of normalization, binary (0,1), positive (range between $0 \dots 1$), and negative (range between $-1 \dots 1$). The earliest investigation showed that for CNN negative normalization is more suitable. Show the all data during train doesn't make sense. During training the model has to see only part of data. The ratio between the train set, the valid set, and the test set is defined as 62/28/10 % respectlivy.

III. Theoretical part

Linear regression is the simplest model to classify images. In linear regression, each pixel has its weight to classify the image. If the weight of the pixel is almost zero, then it doesn't contain important information to classify the image. Figure 5 shows how linear regression deals with digits and letters. However, it creates a problem. For digit 0 and letters O the linear regression has the same way of classifying.

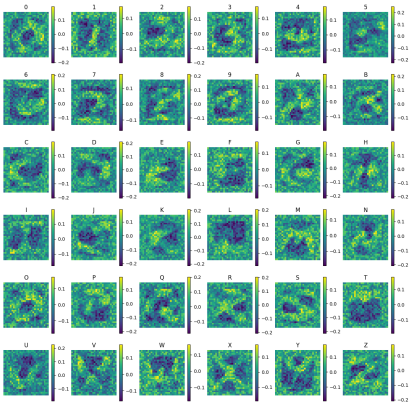

Fig. 5: Weight map of linear regression

Linear regression is very good for simple cases. But for more difficult cases we do a chain of decisions. Connect one linear regression to others we will obtain MultiLayer Perceptron (MLP).

However, the large MLPs usually are overparameterized. It approaches us to the idea of CNN. The idea to use a CNN was a solution to solve a dilemma between a small NN and a large NN [14]. Convolution layers are based on cross-correlation [7]. The goal of the convolution layer is to prepare data for the classifier layer. To do that it uses filters. The right filter has to highlight only important information. But there is the problem, we don't know which filter we have to use. Finding the best filters for convolution layers is the goal of the training process CNN. Using a Deconvolution neural network (DeConv NN) can help to get intuition [5]. Figure 7 shows that convolution layers exactly do. We can compare CNN and human vision. When the child is born, he/she can't classify anything (or his/her filters aren't trained). As a child grows then he/she better recognizes digits and letters. The learning process for a child is dependent on the environment (optimizer), teachers (loss function), and home language (number of classes). It is already known that for example, Russian speakers can better recognize colors than English speakers [15]. Just because Russian vocabulary contains more words to describe colors (more classes). The same problem is with NN. The NN can't give us the output "I don't know that is". Because of vocabulary of the output layer doesn't contain a class for that.

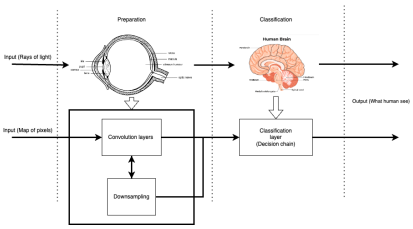

Fig. 6: Comparing Biological NN and CNN

Fig. 7: Visualization of convolution layers [5].

Recognizing handwritten digits and letters is an already-known topic in ML and DL. MNIST dataset which contains 60000 handwritten digits for train and 10000 for testing is the most popular dataset to start studying in image classification. One of the reasons for using the MNIST dataset is that there aren't any hidden details. We can observe how data goes through NN or CNN without complexity architecture. On the website [2] Nielsen, M. A. explains the principles of ML and DL and shows how to implement them from scratch. To train the model he uses the MNIST dataset. However, building everything from scratch is a very complex task, and in most cases, performance will be very poor. Because of that framework, PyTorch was used to implement the approach. In the book [13] is shown a practical approach to how to work with PyTorch. The author goes through particular topics of deep learning and shows how to implement it on PyTorch. There is another popular framework for machine learning which is called Tensorflow. My investigation shows that using PyTorch is the better way to implement its approach. Figures 8 and 9 show that the TensorFlow model has a better loss function. On the other hand, accuracy is the same and after 10 epochs the model is overfitted. It means that is not a real learning process, it is just memorizing the dataset. The goal is to avoid memorizing and try to learn the model.

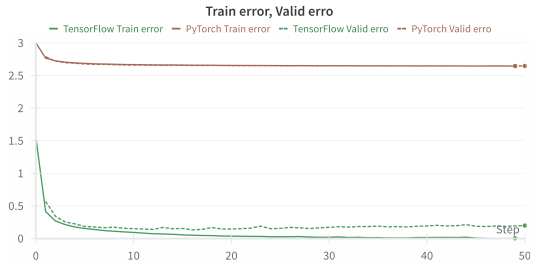

Fig. 8: Loss function. Comparing PyTorch and TensorFlow

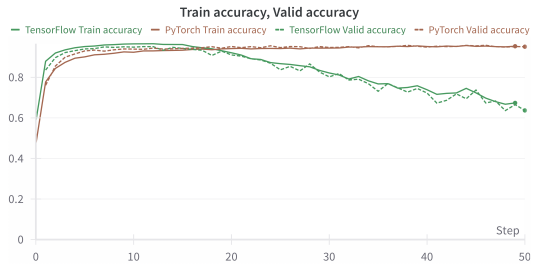

Fig. 9: Accuracy. Comparing PyTorch and TensorFlow

In 1995 Y. LeCun et al. proposed the LeNet-5 in the Bell Laboratories [14]. They compared different learning algorithms to classified digits from the MNIST dataset. Their experiment shows that the linear classifier method has an error of 8.4 % and the Nearest Neighbor classifier (kNN) with n = 3 performs 2.4 %. LeNet-5 which is proposed in [14] is based on the previous architecture LeNet-4. One of the aspects of using CNN is that Linear NN and MLP assume that each value in the input vector is independent. It is already a wrong assumption for image processing. Convolution layers make pixels dependent on their neighbors [14]. When we use convolution layers we have to keep in mind that the shape of the output will be changed. The shape depends on kernel size, padding, and stride. Using the equation which is proposed in [6] it is possible to obtain the desired shape of output.

IV. Implementation

For the supervised method, in general, the training algorithm is the same and independent of which model is used. The performance of the model depends not only on the architecture of the model but also on hyperparameters. As loss function was taken CrossEntropy loss function. It is possible to use MSELoss or BCELoss, but CrossEntropy is a more convenient way and does not require a redesigned target vector.

Choosing an optimizer is usually a difficult question. The two most popular optimizers were taken to approach a final result. Stochastic Gradient Descent (SGD) is the vanilla way to train the model. However, SGD isn't so good when the input to the model is an image. For working with images Adam optimizer is more suitable [17]. The implemented version of Adam in PyTorch gives the ability to switch typical Adam optimizer to AMSgrad mode [18], [7]. But for this approach, only the classical Adam optimizer was used. Tuning of the

optimizer is the second aspect of obtaining good performance. SGD optimizer has two options for tuning. Changing the learning rate and using momentum. Using momentum can improve the loss function, but it also means that the training process has to be longer. At the start, momentum prevents the model from learning fast, but in the end, it will oscillate around the minimum point of the loss function. Because of that momentum was set to zero. The learning rate is an important parameter for Adam as well. In PyTorch, Adam optimizer has already a predefined learning rate which is equal to 0.001 [7]. In the case of SGD optimizer set of learning rates have been testified. The formula to change the learning rate can be described as $lr(i) = 0.001 \cdot e^i$, $i \in [1 \dots 5]$.

Authors of [17] recommend choosing β_1 and β_2 to satisfy condition $(1 - \beta_1) = \sqrt{1 - \beta_2}$. In the case when $\beta_1 > \beta_2$ the ratio $m^*/\sqrt{v^*} > 1$ and the learning rate in the beginning of training will be higher that in the end. Otherwise, the learning rate will be smaller than even the default value. It creates a possibility to obtain the schedule of the learning rate.

The amount of memory in the machine is limited. Training the model on the whole dataset in one epoch is desired, but it requires a lot of memory. Instead of that the dataset can be split into smaller batches. There isn't any formula that describes how big the batch size has to be. The big batch size is good. The model has a better view of data. However, the optimizer makes fewer steps to improve a model in one epoch. The small size means that the view of data is narrow and the model updates the weight for particular samples.

To find the best hyperparameters the model for testing was taken linear regression. Linear regression corresponds to one linear layer and sigmoid function in PyTorch. The timeline to train the model is 50 epochs. It is more than enough to decide does something have to be changed or not.

In the case of MLP, there are a lot of options on how to design the model. However, it has been already said that large MLPs are usually overparameters. To avoid this case, the MLP model contains only three layers. We are free to choose how big the middle layer has to be. It is necessary to find the smallest size to obtain the smallest number of parameters. The range between 250 and 260 neurons has been taken as finding optimum.

After finding the good MLP architectures (or other words the brain of the model) we need to build the preparation part. In the case of image classification as preparation part is more suitable is convolution structures. The images in the dataset aren't complex. Two convolution layers have to be enough for the preparation part. Figure 10 shows the architecture of CNN. The convolution layers aim to save the shape of the input. The kernel size of each layer is matrix 3 by 3. To save the shape of the input, padding is equal one was added [6].

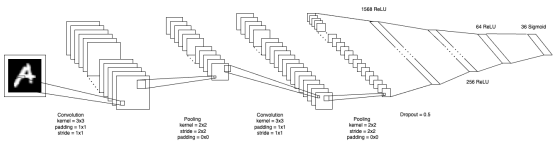

Fig. 10: Diagram of CNN architecture.

Another aspect of the preparation part is downsampling. Downsampling is a necessary part of the preparation part. There are two ways to implement that. Average pooling and max pooling. After pooling the layer desired shape of the output has to be twice smaller. It means that there is only one configuration for the pooling layer (kernel size is matrix 2 by 2, stride is equal to 2, and padding is equal to zero).

For regularization was also used valid loop and dropout after the preparation parts. The valid loop is used to understand when the model reaches the maximum ability to learn. The dropout between the preparation part and the classification part is set to 50 %.

To measure the success of training the model accuracy on the test set is the main measure parameter. Additional parameters such as precision, recall, and F1-score are used to find particular problems in architecture.

To implement the approach in the figure 21 two same CNN architectures (figure 10) have been taken. The main CNN trains on the whole dataset, even on class O and 0. The sub-CNN trains only on classes O and 0. The training process is going independent. The test loop is built on the base of the algorithm in the figure 21.

V. Evaluation

As figure 11 shows Adam optimizer has better performance than SGD optimizer. Keeping going experimenting with the learning rate on SGD optimizer can improve the performance. But Adam optimizer was in the default configuration, which means "Just define and use".

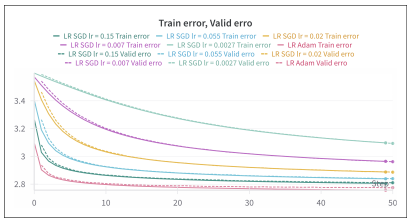

Fig. 11: Loss function. Comparing SGD and Adam.

Figure 12 shows that the difference between the learning rate of 0.055 and 0.15 is very small. Validation accuracy is the same, but the loss for the learning rate 0.15 is smaller. It can be the same issue as in the case of TensorFlow when the model just memorizes the samples. Changing the ratio between the train set, valid set, and test set can illuminate this issue.

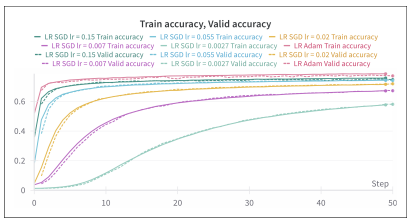

Fig. 12: Accuracy. Comparing SGD and Adam.

After testing the model with Adam optimizer performs the final accuracy of 77 %, but SGD performs only 75 %. But

linear regression has a lot of issues between classes as shown in figure 13.

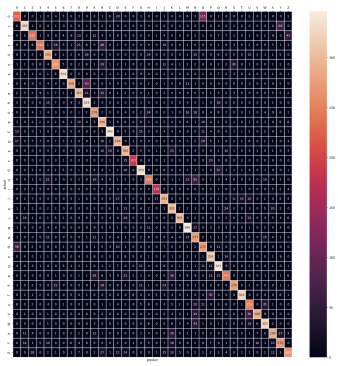

Fig. 13: Confusion matrix for linear regression. Adam. batch size 512.

Another aspect that can improve performance is batch size. However, figures 14 and 15 show that batch size affects mostly only on speed of learning the model. In the end, the model with different batch sizes reached the same accuracy and loss value. But to reduce memory requirements and the number of epochs batch size 512 was taken.

Fig. 14: Loss function for different batch size

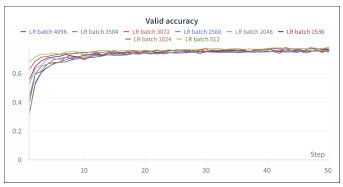

Fig. 15: Accuracy for different batch size

In the case of MLP, experiments show that adding the activation function is necessary. Adding the ReLU function in the input layer and hidden layer gives an average accuracy of 93%. Without activation function accuracy on the test set is a maximum of 69 %. It is even worse than linear regression with a sigmoid function.

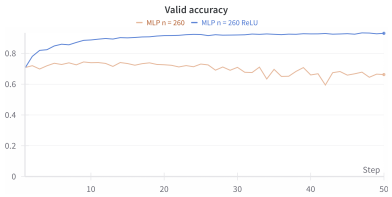

Fig. 16: Accuracy. Comparing MLP with ReLU and without

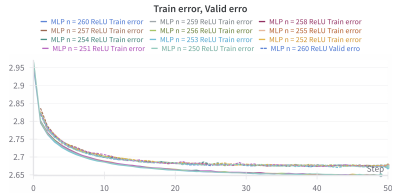

Fig. 17: Loss function for MLPs.

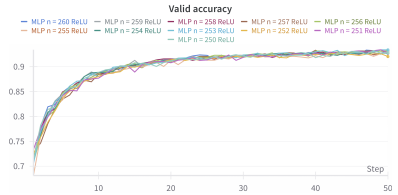

Fig. 18: Accuracy MLPs.

Another experiment shows MLP with ReLU and the size of the middle layer in the range of 250...260 is already overparameters. From the beginning of training, there is overfit. It is what Y. LeCun et al. have tried to avoid in [14]. But accuracy is pretty good. In the end, all models have an accuracy of around 93 %. Since there is no difference makes sense to take the model with the smallest middle layer. If MLP with 3 layers is already overparameters, then there is no reason to make it larger. To improve the accuracy we have to use CNN. The dilemma between average pooling and max pooling is still not solved.

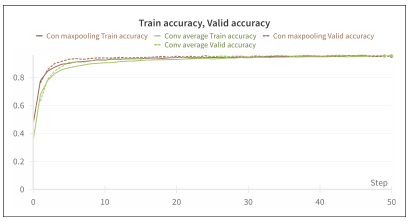

Fig. 19: Accuracy CNN. Comparing max pooling and average pooling.

Experiment with architecture in figure 10 shows that average pooling performs better results. Yes, accuracy and loss are almost the same. However, the confusion matrix shows that average pooling has less misclassification with classes O and 0. In most cases, it classifies the letter o as digit 0. Only 4 cases when it classifies digit 0 as the letter O (2%).

Another convolution architecture is LeNet-5. As it was noticed before this model is designed especially for the MNSIT dataset. Nevertheless, the final accuracy is pretty high (93 % on the test set). However, this CNN is more complex than MLP before, but the accuracy is the same.

After testing, it was found the approach in figure 21 can improve the result for the confusion matrix. The testing result of sub-CNN is equal to 90 % and the testing result for the algorithm in figure 21 is equal to 97 %. Improvement isn't big (only 1 %), but if we look into the confusion matrix, then it is clear that the algorithm works very well. Unfortunately, there

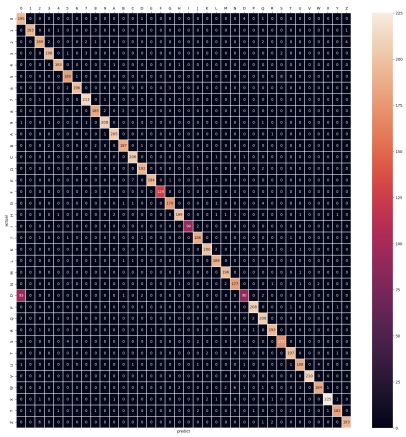

Fig. 20: Confusion matrix for CNN with average pooling.

is a disadvantage to using this method. Main CNN sometimes can predict D, C, 9, and so on as 0 or O. The sub-CNN doesn't even know about that. It can predict only O or 0. It means when the main CNN predicts 9 as O, then sub-CNN still can't solve this dilemma. To solve this issue additional information such as the sample is from word can help. If we already know that the sample is from the word, then digits can't be here, it reduces the number of probability to only 26 classes.

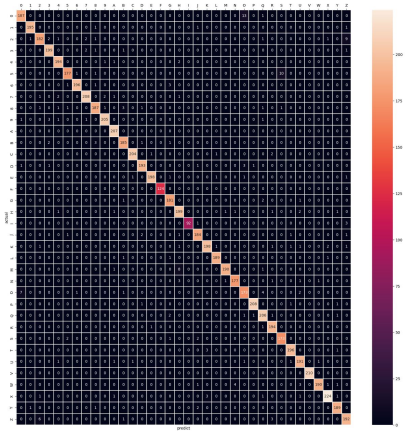

Fig. 21: Confusion matrix for main CNN and sub-CNN.

VI. Conclusion

The project aimed to understand better how ML and DL techniques work. These techniques are very powerful tools. The simplest linear regression already has 77 % accuracy. Adding another layer can improve accuracy, but for MLP there is a limitation, especially in image processing. While experimenting it was shown that MLP may be easily over parameters and it can't reach an accuracy of 100 %. It is necessary to add a preparation part before classifying the sample. Adding the preparation part NN starts to work similarly to how people deal with image recognition. As it was shown artificial NN works similarly to biological NN.

However, the main issue with NN, that it can't say "I don't know" isn't still solved. Another problem is that if the input is rotated it also can't recognize the object. Always the output from NN is just probability for a certain class.

The potential of NN is huge. Human's abilities are limited. The NN aims to find the relationship between the input data and the output. There are a lot of applications of NN from classification images to finding the effect of pills on the disease.

As next step to keep going this topic is trying to improve accuracy and experimenting with cascade connection between NNs (figure 21). Another improvement is building the model for image segmentation and connecting to this network. After that, it will be able to read a text. After converting text to digital text Recurrent Neural Network can predict the meaning or execute the command to do something (depending on field of application). Using a transformer can translate text on paper to another language.

VII. Supplementary material

Supplementary material belonging to this project can be accessed as follows:

- Dataset A-Z letters: https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format
- Dataset MNIST: https://www.kaggle.com/datasets/oddrationale/mnist-in-csv
- Notebook: https://colab.research.google.com/drive/1cpwmFtLppLybieUOR23ix1RjyIcaarc7?usp=sharing
- Logs and graphs: https://wandb.ai/mlteamaau/Machine_learning_project?workspace=user-qfaithencz
- Trained model: https://drive.google.com/drive/folders/1ov4qTaIlO3H6vfHP4L7MYySlHVnWXrfU?usp=share_link

References

[1] RASCHKA, Sebastian a MIRJALILI, Vahid. Python machine learning: machine learning and deep learning with Python, scikit-learn, and TensorFlow 2. Third edition. Expert insight. Birmingham: Packt, 2019. ISBN 1789955750.

[2] Nielsen, M. A. (2015). Neural networks and deep learning. http://neuralnetworksanddeeplearning.com/

[3] Handwritten character recognition using machine learning. (2023, May 12). IEEE Conference Publication — IEEE Xplore. https://ieeexplore.ieee.org/document/10182858

[4] Handwritten text recognition using machine learning and deep learning. (2023, April 6). IEEE Conference Publication — IEEE Xplore. https://ieeexplore.ieee.org/document/10142716

[5] D. Zeiler, M., & Fergus, R. (n.d.). Visualizing and Understanding Convolutional Networks. Dept. Of Computer Science, New York University, USA. https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf

[6] Dumoulin, V. (2016, March 23). A guide to convolution arithmetic for deep learning. arXiv.org. https://arxiv.org/abs/1603.07285

[7] PyTorch documentation — PyTorch 2.1 documentation. (n.d.). https://pytorch.org/docs/stable/index.html

[8] Ronneberger, O. (2015, May 18). U-NET: Convolutional Networks for Biomedical Image Segmentation. arXiv.org. https://arxiv.org/abs/1505.04597v1

[9] A-Z Handwritten Alphabets in .csv format. (2018, February 16). Kaggle. https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format

[10] MNIST in CSV. (2018, May 19). Kaggle. https://www.kaggle.com/datasets/oddrationale/mnist-in-csv

[11] Gradient-based learning applied to document recognition. (1998, November 1). IEEE Journals & Magazine — IEEE Xplore. https://ieeexplore.ieee.org/document/726791

[12] Papers with Code - SpinalNet: Deep Neural Network with Gradual Input. (2020, July 7). https://paperswithcode.com/paper/spinalnet-deep-neural-network-with-gradual-1

[13] Pointer, I. (2019). Programming PyTorch for deep learning.

[14] Lecun, Y., Jackel, L., Bottou, L., & Vapnik, V. (1995). Comparison of learning algorithms for handwritten digit recognition. ResearchGate. https://www.researchgate.net/publication/216792818_Comparison_of_learning_algorithms_for_handwritten_digit_recognition

[15] Winawer, J., Witthoft, N., Frank, M. C., Wu, L. M., Wade, A. R., & Boroditsky, L. (2007). Russian blues reveal effects of language on color discrimination. Proceedings of the National Academy of Sciences of the United States of America, 104(19), 7780–7785. https://doi.org/10.1073/pnas.0701644104

[16] Lhg-Admin. (2019, March 26). How does the human eye work? NKCF.org. https://nkcf.org/about-keratoconus/how-the-human-eye-works/#:~:text=The%20lens%20focuses%20light%20through,optic%20nerve%20to%20the%20brain.

[17] Kingma, D. P. (2014, December 22). Adam: A method for stochastic optimization. arXiv.org. https://arxiv.org/abs/1412.6980

[18] Reddi, S. J. (2018, February 15). On the Convergence of Adam and Beyond. OpenReview. https://openreview.net/forum?id=ryQu7f-RZ