Convolutional Neural Network for Fashion Images Classification (Fashion-MNIST)

Tang Jian Shiun
School of computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
tp068048@mail.apu.edu.my

Tey Jia Yi
School of computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
tp068626@mail.apu.edu.my

Pu Jun Yu
School of computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
tp064307@mail.apu.edu.my

Por Jia Xin
School of computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
tp062856@mail.apu.edu.my

Voon Pei Yi
School of computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
tp063378@mail.apu.edu.my

Zailan Arabee Abdul Salam
School of computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
zailan@apu.edu.my

***Abstract* — Recognizing and classifying images is a significant research topic in the widely used computing technology nowadays – the computational vision. The common ways for classifying image and performing recognition tasks depending on deep learning such as the Convolutional Neural Network (CNN). With the high impact resulting from Artificial Intelligence identified through the transformation in the fashion and apparel industry. It has then been realized that difficulty has been found in terms of understanding the work performed in the industry. In this research, it is aimed to focus on identifying the parameters that are able to affect the accuracy of the particular trained model for fashion image classification using deep learning in neural networks such as CNN with the fashion MNIST dataset.**

Keywords—(CNN) Convolutional Neural Network, (ANN)-Artificial Neural Network, tiny (VGG) Visual Geometry Group, artificial intelligence, clothing classification.

I. INTRODUCTION

Fashion in society nowadays places a stronghold position as most of the population in the globe has a certain idea and practices with fashion. Unique apparels are able to reflect self-concepts as well as lifestyles, indicating a change and showing different times and places. Obtaining the visual classification of clothing products with the extraction from computer vision plays an important step in the fashion industry. The automation in classifying garments based on features allows both producers and data experts to have awareness of general overall production, which is fundamentally important to avoid duplication of products, organization, categorization etc. which smooths the flow and development process.

The MNIST dataset from Zalando's research is used to train models with sets of training and test samples. The purpose of this model is to help with classifying the types of fashion clothing which categorizes into T-shirts, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags and ankle boots as the assigned labels. This model is able to help with the usage of classification for clothes recycling or clothing recognition for quick online shopping. It also serves the general purposes of market research analysis as well as evaluating fashion trend collections. Different approaches to perform visual classification were used starting from image processing to machine learning such as feature extraction, image recognition and template matching etc.

A. Literature Review

Based on the works of (Ng et al., 2023) which analyse the usage of CNN for classifying the images of fruits that is merely like the fashion classification. They have used (DCNN) which is known as the deep convolution neural network which requires the high calculation of restricted conditions hence denying the utilization of DCNN. They have implemented the ANN algorithm to assist in fruit image classification with pre-trained images. The ANN algorithm used in the study was trained on the variation of fruit images from the fruit datasets for identical features. The hyperparameters that they modified were learning rates, epochs and the variation of the activation function. At the end of their study, they have obtained and identified that the optimal option for learning rate is the default at 1e-5 despite 1e-3 allowing the trained model to be more optimal, but much more time for the training process is required.

In the research of (A.Vijayaraj et al., 2022) similar usage and datasets were used which is to perform deep learning in classifying images using the MNIST dataset provided by Zalando. ANN and CNN are both implemented in the stated study. In addition to our current study the tinyVGG which is the Visual Geometry Group, has been implemented for improvements on the study clothing classification. In Vijayaraj's work, it has been found that CNN performs at a better rate with the tested accuracy of 0.9452.

Based on the study of (Lead et al., 2021) which implemented the architecture of different CNN models which are GoogLeNet, MobileNet v2, ResNet-50, ResNeXt-50 as well Wide ResNet-50 with the MNIST dataset for handwritten digit recognition. The aim of the study is to propose an architecture with faster and higher accuracy results. Based on the findings the model of Wide ResNet-50 has obtained the lowest Top-1 error at the result of 0.5278% and Top-5 error of 0.0079% while MobileNet v2 has the fastest training time among the models at 498 seconds (about 8 and a half minutes). Other than the MNIST datasets, the study also

experimented on CIFAR-10 datasets for further research on complex data.

The addition of factors that may contribute to the inaccuracy of the CNN outcomes is based on the research of (Luca et al., 2019) which studies feature extractions in images for fashion product classification. It has been found that misclassification may happen due to the lack of feature displays caused by the solidity of the colours of the clothing. As well as another study based on the analysis of convolutional neural networks for image classification from the works of (Neha et al., 2018), has been founded that CNN models tend to get confused with live-tested objects than static which adds on that the complexity of frames from the real-time data are able to cause confusion to the network layers.

II. BACKGROUND

A. Image Database

The dataset used for our experiment is Fashion-MNIST provided by Zalando Research and is open for public download on Kaggle.com or GitHub.com. The Fashion-MNIST dataset consists of approximately 70,000 images of fashion products with 28x28 grayscale images of 10 distinct fashion labels. The labels include T-shirts/tops, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. A dataset of 70,000 images was divided into two parts, with 60,000 images used as training data. The remaining 10,000 images were utilized as testing data for evaluating the accuracy of the trained model (Yamazaki, 2018; Zalando Research, 2020). The dataset is widely used in the AI/machine learning community to build and test computer vision models. It can also be used to benchmark the performance of various AI algorithms. The Google Collab platform will be utilized in our experiment. To enable easy access to the downloaded dataset, the dataset is loaded onto Google Drive, as both platforms can be integrated alongside each other.

B. Image Classification

Image classification is a supervised learning problem in which defines as a collection of target classes (entities to recognize in pictures) and training models to identify them using labelled images. A classification algorithm uses an image as input and predicts which class it belongs to depending on its features (Papers with Code, 2011; Google, 2022). Artificial Neural Networks (ANNs) are limited in their ability to handle spatial structures, making CNNs a preferred method for image classification. In image classification, ANNs treat each pixel independently, hence resulting in poor spatial reasoning results. CNNs are composed of convolutional layers that extract features like edges and textures; pooling layers and then down sampling the feature maps to improve detail captures. To perform the final classification, the fully connected layer flattens the spatially organized feature maps, considering the spatial arrangement of the features. By backpropagating through shared-weight convolutional layers, overfitting can be reduced and efficiency increased. As a result, CNNs can detect patterns, recognize relevant features, and exploit spatial redundancy while requiring fewer parameters (Meel, 2022; Sharma, 2023). When it comes to challenging tasks like image classification, CNN outperforms ANN. Therefore, fashion classification tasks can benefit from CNNs despite requiring substantial training data due to the high number of parameters and high computational power.

III. ALGORITHMS AND APPROACHES

A. Artificial Neural Network

Neural networks, also known as artificial neural networks (ANNs), are a subclass of machine learning that serves as the basis for deep learning approaches. With pre-trained image datasets downloaded from Zalando's article images, the ANN method was used to help Fashion MNIST classify the images. The source code used in this research belonged to mrdbourke from Github and was written in Python. The design was influenced by the way organic neurons communicate with one another in the human brain (IBM, n.d.).

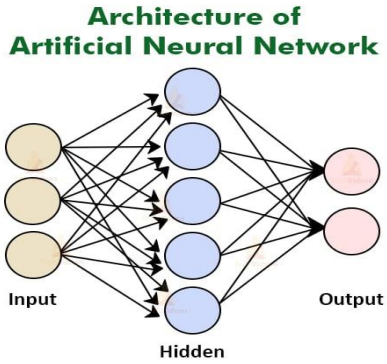

Fig. 1. Architecture of artificial neural network

An input layer, one or multiple hidden layers and an output layer make up the node layer of an ANN as shown in Fig 1 (Upadhyay, 2023). Each node in ANN is interconnected with a weight and threshold. A node is activated and contributes data to the uppermost layer of the network if its output exceeds the defined threshold value (IBM, n.d.). If the output does not exceed the defined threshold value, data will not be sent to the network's next tier. To develop and improve accuracy over time, training data is crucial for neural networks.

In this research, the ANN algorithm was used to train various fashion apparel from the dataset and classify using attributes that were the same in each image. The training results were produced as a table to display the accuracy, loss and training time level after 5 epochs for each batch of images. A confusion matrix is also used to see the accuracy of the result of the training.

B. Convolutional Neural Network

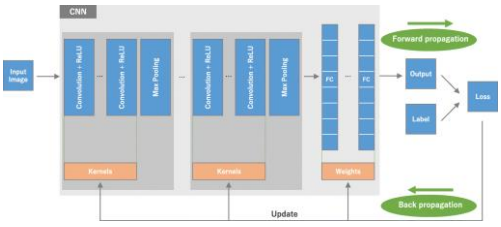

Fig. 2. Architecture of convolutional neural network

Convolutional Neural Network (CNN) is a type of Artificial Neural Network that focuses on processing data for two-

dimensional graticule images. CNN is composed of multiple mathematical operations, known as the convolution layers which have specialized linear operations. In a digital image, every feature represents a pixel value that is stored in a two-dimensional (2D) grid, or array of numbers. To extract the optimizable feature, a small grid of parameters called the kernel is applied at each image position as shown in Fig 2 (Yamashita et al., 2018). The input data that needed to be trained by the CNN model's architecture will be implemented with weights and biases. This is to differentiate different elements of the images to make them stand out from one another. The output of the CNN model is known as feature maps that are shown in arrays. This makes CNNs extremely effective for image processing (Saha, 2018).

This study aims to use CNN algorithm model to uncover key information in image data of the clothing image datasets by processing images, performing classification, segmentation and object recognition. To achieve this objective, the CNN model was given many clothing images prepared by Zalando research to identify the numerous patterns present in each image and adjust the bias and weight of the nodes. The RGB colour of the photos is first turned into grayscale to simplify algorithms and as well eliminate the complexities related to computational requirements, and then the image representations are chosen and altered to facilitate the training process (isahit, n.d.). Convolutional, pooling and totally connected layers are the three layers that will be employed in CNN. These levels each apply a different operation to the incoming data. Features from the input image are extracted using filters. Feature extraction and classification must be done to meet the research's goal. The totally interconnected layer collects information from feature maps and produces the final categorization (Kadam et al., 2020).

C. TinyVGG

Convolutional neural networks (CNNs) that are used for image processing and computer vision employ a sort of architecture called TinyVGG. It is a scaled-down version of the conventional deep CNN architecture, the VGGNet, which has many layers (Boesch, n.d.). The goal of TinyVGG is to be more effective than the original VGGNet while maintaining its strength for a variety of application scenarios. Convolutional layers in the TinyVGG design are completely linked, which means that each neuron is coupled to every other neuron in the layer below (CNN Explainer, n.d.). Multiple convolution layers are stacked in the architecture, however in shallow TinyVGG, just two sets of four convolution layers are typically included. One of TinyVGG's key qualities is the utilisation of all 3x3 filters (Sucky, 2023).

According to CNN Explainer (n.d.), the characteristics that distinguish different images from one another in the convolution layers are extracted by the learnt kernels (weights) to form the basis of CNN. It will show connections between the convolutional layer and the preceding layers as working with the convolutional layer. The output or activation map of the current convolutional neuron is produced by the convolution process using a unique kernel that is represented by each connection. (CNN Explainer, n.d.). The previous layer and a distinct kernel are combined in an element-wise dot product by the convolutional neuron to produce an appropriate neuron. These distinct kernels will provide an equal number of intermediate outcomes. The total of all the intermediate findings plus the learnt bias yields the convolutional neuron. (CNN Explainer, n.d.). Hyperparameters inside the convolutional layers are included below.

1. **Padding** is often necessary when the kernel extends beyond the activation map. It enables an architectural designer to create deeper networks by maintaining the spatial scale of the input. Padding will add a border at the boundaries of activation maps, leading to superior productivity. (CNN Explainer, n.d.).
2. **Kernel size,** also known as the filter size, refers to the dimensions of the sliding window over the input. The image classification job is significantly impacted by the choice of this hyperparameter. For instance, lower kernel sizes can extract from the input a substantially greater amount of data including extremely local characteristics. A lower kernel size also results in a lesser drop in layer dimensions, allowing for a deeper architecture. However, a high kernel size extracts less data, which causes a rapid drop in layer dimensions and frequently results in lower productivity. Larger features can be extracted more effectively from big kernels. (CNN Explainer, n.d.).
3. **Stride** value specifies how many pixels the kernel should move over each time. For instance, Tiny VGG employs a stride of 1 for its convolutional layers. This implies that the dot product is done on a 3x3 window of the input to generate an output value, then is moved to the right by one pixel for every subsequent operation. Like kernel size, stride influences a CNN. More features are learnt when the stride is shortened since more data is extracted, which also results in larger output layers. (CNN Explainer, n.d.).

Following that, the neural network will be developed layer by layer by stacking hidden layers one after another thanks to the model's sequential routing of its layers. The deep neural network was altered using the sequential technique in order to improve task recognition and execution. A flattening layer, a dropout layer that prevents model overfitting, and a dense layer that acts as the output layer and uses ReLU as the activation function to aid with multi-class classification were the layers' parameters of the model. Lastly, the entire process will be compiled and trained to evaluate the model's performance. The loss and accuracy will be indicating the model's performance.

IV. ALGORITHMS IMPLEMENTATION

A. Purpose

The raw clothing images are usually too large for the neural network to compute. As mentioned above, the amount of computation time and resources are abundant amount. Hence,

CNN's mechanism of convolving and pooling the data while transforming the data into a smaller set of tensors. This saves a large amount of time by reducing the number of inputs for the neural network which indirectly decreases the computation resources maintaining the accuracy. As a result, more resources for the experiment could be spent at the focus of other aspects.

B. Environment Setup

With the dataset of clothing images from Zalando research, the problem is approached with a model that is structured with two two-dimension convolutional layers. The model is programmed in Python which is mentioned in the section above. As Google Colab provides cloud resources, the model will be trained through its provided Cuda GPU and if not the virtual machine's CPU. In this experiment, the GPUs are available as shown in Fig 3. By using the same hardware with identical specifications, the factor of the inconsistent model performance to hardware issues could be negligible. Thus, the environment for experimentation regarding the model is better.

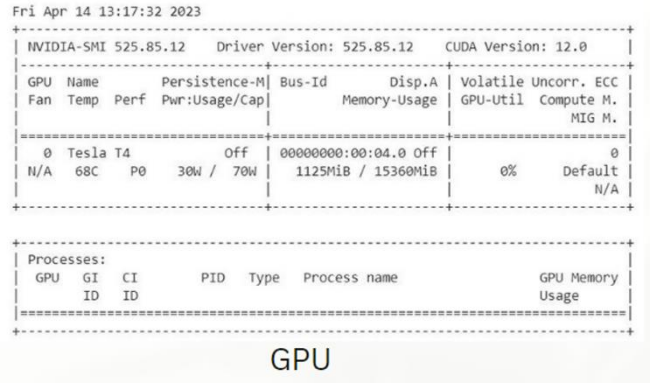

Fig. 3. Google Colab GPU specification

C. Model Structure

In this scenario, the TinyVGG architecture will be implemented for CNN. This structure could be achieved with the help of the Pytorch library which consists of the functions that are needed for this experiment. The layers of convolved data will be passed into the pooling layers. Since the MaxPool algorithm is used, the largest value along the matrix with a size of two by two will be picked out to make the feature map. This eventually creates a more abstract version of the feature by acquiring the necessary data. Hence, it will result in using more reasonable computation resources.

In the neural network, every node will have the function to determine the transmission of signal over the other nodes by a threshold. This function is known as the activation function. In this CNN, the activation function used is the rectified linear unit function (ReLU). The visualization of the graph is plotted as Fig 4. As output, the input will range from 0 to the input value provided it is positive. The function provides a range of positive outputs that is proportionally different and not all near zero-values that is seen in other activation functions such as the Sigmoid Function. As long as the output is not 0, the nodes transmit the next signal. Aside from this, the other aspects such as epochs, learning rate, pooling layer, loss function and optimizer function.

$$R(z) = \begin{cases} z & z > 0 \\ 0 & z <= 0 \end{cases}$$

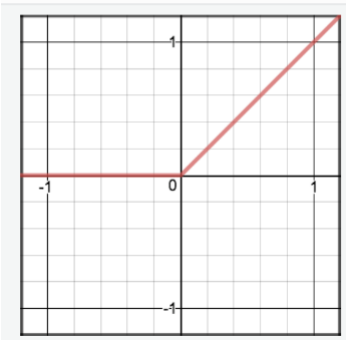

Fig 4. ReLU function.

D. Parameters and Functions Modified

As the default model for comparison, several parameters remain constant unless the parameter or function is being experimented on. To note, the default model is set to have a 3 epoch, batch size of 32, learning rate of 0.1 with MaxPool function for pooling layer, CrossEntropyLoss for loss function and SGD as optimizer function.

Epochs are known as the number of times that the model has forward pass and backpropagated the data through the model. If a training sample is a thousand in size, one epoch is the parameter to measure the amount of per forward pass and backward propagation for all 1000 samples. In this process, the training data will be trained on the network. Which theoretically let the performance of the model increase after each epoch. In the experiment, different numbers of epochs will be used to verify the amount of difference in training loss and accuracy differs.

Having many epochs may potentially increase the performance of the model. The downside to this is that the computation power needed to process all training samples in one iteration is enormous. Hence, the role of another parameter which is batch size may solve this. The batch size refers to the number of samples that are used for one iteration. If the total training sample is a thousand in total, a 500-batch size will take two iterations to finish one epoch. By that, a few batch sizes will have experimented with a constant epoch of 3 in this test.

Another layer of the model for feature extraction includes the pooling layer. The former pooling layer is MaxPool which extracts the max value out of the feature map. This function takes the value that is max out of the pool size matrix. For the experiment, the pooling layer will be formed using the AveragePool function. This function will eventually take the average value of the pool.

After having the features extracted in the pooling layer, the loss function comes to play to evaluate the overall accuracy loss of the model. Based on the loss function, an overall loss will be calculated between the output and targeted value. In the default model, the cross-entropy loss function is used. In the experiment, other loss functions such as MultiMarginLoss and NLLLoss functions will be used.

The learning rate is the amount that the optimizer adjusts the weights of the model. This parameter is constant when paired with optimizer functions such as SGD which is in the

default model. The smaller learning rate could mean that the model converges in small steps but could end up in local minima. Hence, different learning rates are used in the experiment.

Lastly, the optimizer function plays the role to tune the weights to optimize the model. In the default model, SGD is used as it works with a constant learning rate. In the experiment, the Adam optimizer function will be used. As this optimizer uses an adaptive learning rate as time goes by, it will be able to optimize the model without being stuck in a local minimum.

V. RESULT AND DISCUSSION

After changing different parameters, we obtain all the results of the parameters. We collect all the training loss, training accuracy, testing loss, testing accuracy and time to compare the parameters.

TABLE I shows the result of using different optimizer. There are two different optimizers, one is SGD optimizer and another one is Adam optimizer. Comparing the training loss and accuracy, we can notice that the result of SGD optimizer is much better than Adam optimizer. The training accuracy of the Adam is only 9.79% which is very low. It means that SGD optimizer is more effective to minimize the training loss and improve the model accuracy during training. Comparing the testing loss and accuracy, the result of SGD optimizer is also better than Adam optimizer. Regarding the time, time used by SGD optimizer is a bit lower than Adam optimizer. Based on the observations, we can conclude that in this dataset by using TingVGG architecture, SGD optimizer performs better than Adam optimizer in terms of both training and testing.

TABLE I. RESULTS OF DIFFERENT OPTIMIZER

Optimizer	Training loss	Training Accuracy	Testing loss	Testing Accuracy	Time (seconds)
SGD	0.32377	88.29%	0.33105	87.92%	40.085
Adam	2.31583	9.79%	2.31064	10.00%	42.254

TABLE II shows the results of using different loss functions. From the result, we can notice that different loss functions have different loss, accuracy, and time to train. The result shows that the MultiMarginLoss is the is better than CrossEntropyLoss function and NLLLoss function. It has the highest accuracy and lowest loss. CrossEntropyLoss and NLLLoss have similar model accuracy, but CrossEntropyLoss has a lower accuracy and a higher loss. By comparing the time, we notice that CrossEntropyLoss takes the shortest time, which is 133.081 seconds. The second is the MultiMarginLoss function, it takes 333.40 seconds. NLLLoss function has the longest training time among the three loss functions. In conclusion. We can conclude that if the purpose of achieving the highest accuracy, NLLLoss performs better than the two functions. If a good balance between accuracy and training time function is needed, CrossEntropyLoss and NLLLoss can be the option.

TABLE II. RESULTS OF DIFFERENT LOSS FUNCTION

Loss Function	Model Loss	Model Accuracy	Time (Seconds)
CrossEntropyLoss	0.32341	88.64&	133.081
MultiMarginLoss	0.0336	99.67%	333.40
NLLLoss	0.2578	90.58%	438.27

TABLE III shows the result of using different batch sizes. There are three different batch sizes, which are 8, 32, and 128. Based on the table, we can notice that the loss will become lower if we change the batch size from 8 to 32 but increase a bit when we change the batch size from 32 to 128. Besides that, the accuracy of the training and testing also increases when we change the batch size from 8 to 32 and decreases when we change the batch size to 128. For overall accuracy and loss, batch size 32 has a better performance. For the time taken, increasing the batch size will decrease the time. This is because larger batch sizes only require fewer iterations to process the whole dataset. Therefore, it will increase the speed of training and decrease the time for training.

TABLE III. RESULTS OF DIFFERENT BATCH SIZE

Batch Size	Training Loss	Training Accuracy	Testing Loss	Testing Accuracy	Time (Seconds)
8	0.35989	86.81%	0.35696	86.77%	76.311
32	0.32362	88.23%	0.32483	88.47%	48.381
128	0.39404	85.80%	0.37490	86.82%	32.522

TABLE IV shows the result of using different epochs. There are three different epochs, which are 1, 3 and 21. Based on the table, we can notice that when the number of epochs increases, the training loss and testing loss will decrease. Besides that, we also notice that when the number of epochs increases, the accuracy of the result also increases, but from. Based on the observations, it indicates that more epochs will decrease the loss and increase the accuracy. Therefore, more epochs allow the model to optimize and fit the training data better. In terms of the training time, we notice that the time will increase when the number of epochs increases. It is because more epochs require more iterations over the training data. Therefore, it will require longer training time and increase the time.

TABLE IV. RESULTS OF DIFFERENT EPOCHS

Epochs	Training Loss	Training Accuracy	Testing Loss	Testing Accuracy	Time (Seconds)
1	0.59329	78.43%	0.37961	86.37%	14.180
3	0.31741	88.59%	0.31458	88.82%	53.098
21	0.22457	91.80%	0.28633	89.91%	300.709

TABLE V shows the results of different learning rates. From the table, we can notice that the training loss will decrease when the learning rate increases. This is due to the reason that a higher learning rate can help the model converge faster. Therefore, it will have a lower training loss. Regarding the accuracy of the model, it shows that the result is like the loss, which means when the learning rate increase, the accuracy of the model will also increase. This implies that the higher learning rate can help the model generalize better to the data. In term of the training time, it shows that the training time increases when the learning rates increases. This is

because higher learning rates can cause the model to converge faster, but it requires more iterations. Therefore, it will cause the training time to become longer.

TABLE V. RESULTS OF DIFFERENT LEARNING RATE

Learning Rate	Training Loss	Training Accuracy	Testing loss	Testing Accuracy	Time (Seconds)
0.001	0.60513	78.10%	0.61940	76.78%	105.216
0.01	0.34034	87.80%	0.35986	87.41%	113.379
0.1	0.26431	90.29%	0.29796	89.45%	106.413

TABLE VI shows the results of using different pooling layers. From the table, we notice that the loss for both pooling layers is quite similar. Regarding the accuracy of the model, we also notice that the accuracy for both Maxpool2d and Avgpool2d is also relatively similar with an insignificant difference of <1%. Both pooling layers are suitable for this model since they improve generalizability and produce high-accuracy results while reducing loss effectively, but Maxpool2d requires slightly more training time than Avgpool2. This is likely due to the different computational requirements of the two pooling layers. In conclusion, we can deduce that both Maxpool2d and Avgpool2d show comparable performance in terms of loss and accuracy. However, Maxpool2d training time will be slightly longer than Avgpool2d by approximately 9 seconds.

TABLE VI. RESULTS OF DIFFERENT POOLING LAYER

Pooling Layer	Training Loss	Training Accuracy	Testing Loss	Testing Accuracy	Time (Seconds)
Maxpool2d	0.32373	88.21%	0.32566	88.42%	50.361
Avgpool2d	0.33087	87.95%	0.33071	87.78%	41.623

ACKNOWLEDGMENT

We would like to thank Zalando Research for their generosity in making the Fashion-MNIST dataset available to the public on Kaggle and GitHub, which is essential for our research. Without Zalando, this study would not have been feasible.

REFERENCES

Brendan, F. (n.d.). *Activation Functions — ML Glossary documentation.* https://mlcheatsheet.readthedocs.io/en/latest/activation_functions.html

Boesch, G. (n.d.). *VGG Very Deep Convolutional Networks (VGGNet) - What you need to know*. Viso.ai. https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/

CNN Explainer. (n.d.). Poloclub.github.io. https://poloclub.github.io/cnn-explainer/

Donati, L., Iotti, E., Mordonini, G., & Prati, A. (2019). Fashion Product Classification through Deep Learning and Computer Vision. *Applied Sciences*, *9*(7), 1385. https://doi.org/10.3390/app9071385

Google. (2022, July 19). ML Practicum: Image Classification. Google Developers. https://developers.google.com/machine-learning/practica/image-classification#:~:text=Image%20classification%20is%20a%20supervised

IBM. (n.d.). *What are Neural Networks?*. IBM. *https://www.ibm.com/topics/neural-networks#:~:text=Neural%20networks%2C%20also%20known%20as*

isahit. (n.d.). Why to use grayscale conversion during image processing? (n.d.). isahit. https://www.isahit.com/blog/why-to-use-grayscale-conversion-during-image-processing#:~:text=Why%20is%20grayscale%20needed%20for

Kadam, S. S., Adamuthe, A. C., & Patil, A. B. (2020). CNN model for image classification on MNIST and Fashion-MNIST dataset. *Journal of Scientific Research*, *64*(2), 374–384. https://doi.org/10.37398/jsr.2020.640251

Lead, M. S., Chen, B. B. C., Tan, G. Y., Chai, H. T., & Abdul Salam, Z. A. (2021). MNIST handwritten digit recognition with different CNN architectures. Journal of Applied Technology and Innovation, 5(1), 2600–7304. https://dif7uuh3zqcps.cloudfront.net/wp-content/uploads/sites/11/2021/01/17192613/MNIST-Handwritten-Digit-Recognition-with-Different-CNN-Architectures.pdf

Meel, V. (2022, February 1). ANN and CNN: Analyzing Differences and Similarities. Viso.ai. https://viso.ai/deep-learning/ann-and-cnn-analyzing-differences-and-similarities/#:~:text=ANN%20is%20ideal%20for%20solving

Ng, Y. R., How, Y. H., Cheong, Y. K., Omer, M., & Abdul Salam, Z. A. (2023). Convolutional Neural Network for Fruit Image Classification. Journal of Applied Technology and Innovation, 7(1). https://dif7uuh3zqcps.cloudfront.net/wp-content/uploads/sites/11/2022/12/14090157/Volume7_Issue1_Paper5_2023.pdf

Papers with Code. (2011). Papers With Code : Image Classification. Paperswithcode.com. https://paperswithcode.com/task/image-classification

Saha, S. (2018, December 16). *A comprehensive guide to Convolutional Neural Networks—the ELI5 way*. Towards Data Science. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Sharma, P. (2023, April 13). CNN vs ANN for Image Classification. Www.tutorialspoint.com. https://www.tutorialspoint.com/cnn-vs-ann-for-image-classification

Sucky, R. N. (2023, February 28). *Complete implementation of a mini VGG Network for image recognition*. Towards Data Science. https://towardsdatascience.com/complete-implementation-of-a-mini-vgg-network-for-image-recognition-849299480356

Upadhyay, A. (2023, July 10). *20 Must-Know Topics in Deep Learning for Beginners*. Medium. https://medium.com/@aspershupadhyay/mastering-deep-learning-20-key-concepts-explained-ea405aa6603d

Vijayaraj, A., Vasanth Raj, P. T., Jebakumar, R., Gururama Senthilvel, P., Kumar, N., Suresh Kumar, R., & Dhanagopal, R. (2022). Deep Learning Image Classification for Fashion Design. *Wireless Communications and Mobile Computing*, *2022*(Volume 2022), e7549397. https://doi.org/10.1155/2022/7549397

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional Neural Networks: An overview and application in radiology. *Insights into Imaging*, *9*(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

Yamazaki, N. (2018, January 11). Zalando Engineering Blog - The Faces Behind the Fashion-MNIST. Zalando Engineering Blog. https://engineering.zalando.com/posts/2018/01/faces-behind-fashion-mnist.html

Zalando Research. (2020, November 15). zalandoresearch/fashion-mnist. GitHub. https://github.com/zalandoresearch/fashion-mnist