25 years on: Is LeNet-5 still relevant

An evaluation on LeNet-5 reproducibility

Marc Johnston
UCL Civil, Environmental & Geomatic Engineering Department
ucesmmj@ucl.ac.uk

Thitiwich Thummakul
UCL Civil, Environmental & Geomatic Engineering Department
ucestth@ucl.ac.uk

Siwei Lu
UCL Geography Department
ucfasl4@ucl.ac.uk

Yuri Lai
UCL Civil, Environmental & Geomatic Engineering Department
ucesxla@ucl.ac.uk

ABSTRACT

In this paper, we re-examine Yann LeCun's seminal paper in 1998, "Gradient-Based Learning Applied to Document Recognition," leveraging modern computational advancements to reproduce and extend the original experiments. Our objective is not only to reproduce the results achieved with the MNIST dataset but also to explore the model's performance on two alternative datasets, Extended MNIST (EMNIST) and Fashion MNIST (FMNIST). This endeavour seeks to validate the enduring efficiency of gradient-based learning techniques in document recognition and assess their adaptability to more diverse and complex image recognition challenges. Our findings reveal that the performance from our replicated model falls short of the results achieved in the original paper, using the original MNIST data, after evaluation using statistical hypothesis tests. However, subsequent parameter tuning experimentation did allow us to meet the threshold for accepting comparable levels of accuracy had been achieved. Furthermore, the experiment with the EMNIST and FMINST demonstrated the model's robustness and versatility across different contexts. This research underscores the continued relevance of LeCun's foundational work in the current technological landscape and offers valuable insights into the potential for further advancements in the field of image classification.

In summary we show that despite reproducing the original experiment at the R2 level, it is not always enough to replicate the original results. This only serves to highlight the importance of reproducibility as without it, reproducing and proving the results of any paper is challenging.

Our findings also show that hyperparameter selection is critical, and cross validation should always be considered. Lastly, there is significant room for future work and experimentation; these suggestions are reviewed and detailed.

Keywords

Machine Learning; Convolutional Neural Network; Reproducibility; Computer Vision;

1. INTRODUCTION

This work attempts to reproduce the results from Yann LeCun's seminal work in the 1998 paper ***Gradient-Based Learning Applied to Document Recognition [1].*** *The* significance of this paper needs to be explained within historical context. During the 1980s and 1990s, most cutting-edge pattern recognition systems consisted of a feature extractor highly specific to the type of data it is designed to work with, and a more general-purpose classifier. The limitation was that because the feature-extractors rely on a specific set of heuristics, the system cannot learn from anything other than its intended learning targets.

With progress in hardware capabilities, inventions of new Machine Learning (ML) techniques, and availability of large datasets, automatic ML algorithms like Neural Networks (NNs) are gradually replacing heuristics-based systems. Automatic ML algorithms such as Neural Networks can discover latent patterns and hidden information from data without the type of hardwired and explicit logic previous AI systems depend on.

Despite being more versatile, NNs suffered from the limitations in computing capabilities at the time. They were conceptually possible but difficult to implement. LeCun's personal contribution to this field was introducing and popularizing a more efficient and performant version of Neural Networks --- Convolutional Neural Networks and demonstrated its capabilities in tasks such as handwriting recognition. LeCun's own configuration of CNN (Convolutional Neural Network) architecture - LeNet-5 - achieved remarkable results on the task of recognition and classification of hand-written digits from the MNIST dataset. It was performant but efficient at the same time. Other classifiers such as RS-SVM poly 5 achieved striking accuracy on par with LeNet-5 but required significantly more computational and memory resources, while linear classifiers with substantially less computational requirements could not match its performance even closely [1].

The superiority of CNNs over other contemporary models such as linear classifiers and feed-forward neural networks can be explained by the following: Image data contain complex patterns are not linearly separable, making structurally simple algorithms like logistic regression (that rely entirely on drawing linear boundaries to classify) unfit for the job.

The paper outlines the mechanisms of a CNN that made it more capable than its contemporary models. In comparison to a

conventional feed forward neural net, CNNs are generally more computationally efficient when it comes to image classification. In a fully connected NN, a high-resolution image with several hundreds or even thousands of pixels, flattened would produce a huge number of adjustable weights. CNNs on the other hand address this problem because local pixels share the same weights as the filters scan through the image [2]. Also, by flattening an image into a 1D array as an input, fully connected NNs completely ignore the local spatial correlations inside an image itself, which are often very valuable for making classification decisions. CNNs pick up this information because the feature extractor (filters and kernels) inherently works in a 2D space through the process of convolution. CNNs are also tolerant of local distortions and translation of image objects, which puts it at a great advantage over NNs for hand-writing recognition tasks as everyone has a distinct writing style that produces different looking characters.

This paper is foundational to contemporary deep learning. The rapid advancement of AI systems in the 2010s can be directly attributed to LeCun's work in 1998. Large language models and generative AI models we benefit from today would not have been possible without it.

Machine Learning, like many other scientific disciplines, is facing a reproducibility crisis. It is therefore important to reproduce the results of this paper to re-confirm some of the most fundamental concepts introduced by this paper that underpin deep learning today. Hardware and software differences are one of the biggest hindrances to ML reproducibility, the same algorithms with the same configuration and parameters can produce drastically different results running on different GPUs/CPUs and even compiler settings [3]. The original paper and model by LeCun et al. were proposed before the modern ML ecosystem emerged. High performance GPUs and ML libraries that provide out-of-the-box components to build ML algorithms with did not exist back then. It is therefore important to reproduce the findings of this paper to make sure the outstanding performance of LeNet-5 was not an incidental byproduct of the physical and software environments that powered it.

It is important to evaluate the model against different datasets to validate its performance. A ML model should be able to perform well on new unseen data, instead of simply picking up a set of characteristics specific to a dataset. If it performs consistently well across different datasets that means it has truly internalized the latent information and patterns from its learning targets. We therefore chose two extra datasets; EMNIST, an extended version of the MNIST dataset that includes handwritten Alphabet letters in both upper and lower cases, and FMNIST, which includes greyscale images of clothing items from skirts to boots.

2. METHODOLOGY

2.1 Reproduction Strategy

Gundersen et al. [4] reference three degrees of reproducibility; namely Experiment, Data and Method. The former is much more specific, aiming to replicate results exactly, where the latter is more generalised, aiming only to replicate the method rather than the same data and experiment [3]. This is shown in figure 1.

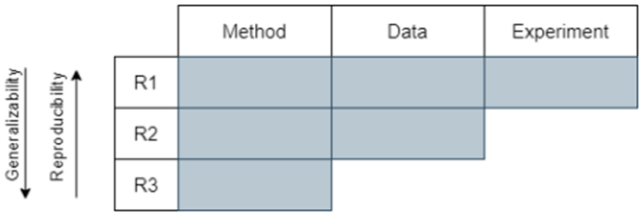

Figure 1 - Degrees of reproducibility. [3]

The aim of reproducing any paper is to achieve every aspect as accurately as possible, meaning R1 in figure 1 is our ideal goal. To achieve this, the same method, data and experiment are required. This is detailed and reflected upon (with regards to our chosen paper) in table 1.

The same *experiment* (hyperparameters and software, including specific versions) as the original paper	Machine Learning (ML) papers seldom include exact details of the software, versions and code meaning a replica can only ever approximate some of these elements of the original experiment. The paper we are replicating [1] does document some hyperparameters (such as the kernel size and learning rate) therefore these have been included where available.
The same *data* as the original paper	The data used in the original paper (the MNIST dataset) has become a de-facto standard (referred to in [5] as the 'hello-world' of AI programming) and as such is widely available, aiding the paper reproduction. It is included in both PyTorch [6] and TensorFlow [7] and is still available on Yann LeCun's website [8]
The same *method* as the original paper	The method is described in the paper in detail and as such can be reproduced using the functionality available within modern data science tools

Table 1 – Reflection on reproducibility

As not all details of the experiment are available (e.g. software) and using [3] as a definition guide, our reproduction will be considered 'R2 – Data Reproducibility'. Any differences in results are likely due to differences in the concrete implementation [3].

2.2 Additional dataset selection

Technology moves fast and as the creator of a popular alternative to MNIST has stated, "MNIST is too easy"; modern CNNs are regularly achieving 99.7% with classic algorithms not far behind at 97% [9]. As such, two additional more modern datasets have been chosen to reflect current data complexities and

determine if the algorithm is still robust, relevant and if it holds up over twenty-five years on.

The EMNIST dataset was chosen for being closely aligned to the original dataset whilst offering a more challenging classification task on which to perform a comparison as it offers lower and uppercase characters as well as digits [10]. It also offers balanced and unbalanced classes which offers to opportunity to test the algorithms' ability to cope with underrepresented data that could introduce bias or poor performance on data with fewer examples.

The second dataset chosen is the Fashion MINIST dataset – something that was specifically designed to challenge and benchmark ML algorithms [11]. This dataset consists of greyscale images of fashionwear, which is very different to the characters in MNIST and EMNIST. It will test the original algorithm versatility by using it to classify something it was never designed for.

Both datasets are available by the authors in the same format and resolution as the original MNIST dataset making them excellent candidates for comparison.

2.3 Algorithm Implementation and Experiment Setup

Our methodology goal for the reproduction is to get as close to the original setup as possible. Fortunately, the authors included considerable detail on the implementation of LeNet-5.

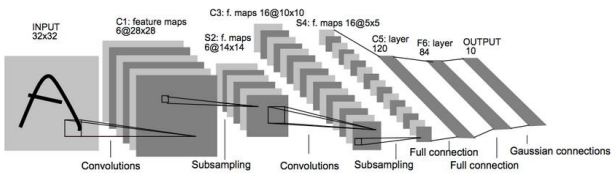

Figure 2 - LeNet-5 Architecture. LeCun et al., 1998 [1]

Two notable items that aid in the replication are the availability of the original dataset and the detailed structure of LeNet-5. As can be seen in (figure 2), the numbers of and types of each layer is specified along with the input resolution and hyperparameters such as the kernel size, stride, and number of feature maps. The MNIST (as well as the EMNIST, FMIST datasets) include predefined test and training splits in separate source files meaning there was no need to create this split – enabling us to reuse the exact images used to train and test the model.

Keras [12] was chosen as our replication tool as an environment that allows developers to build (and fine-tune) all LeNet-5 layers using simple built-in functions. For example, we were able to replicate the custom activation function, scheduled learning rate and custom network initialisation from the paper.

The description of the paper and corresponding functionality implemented, which imitates the original architecture as closely as possible (without having the original software) is shown in Table 2.

LeNet-5 Architecture (from paper [1])	**Functionality Implemented/Parameters**
20 training set iterations	Keras set to run for 20 epochs
Convolution layer 1 (of 3) – **C1** • 6 feature maps (28x28 in size) • 5x5 kernel	Keras Conv2D Layer • Filters=6 • Kernal size=5x5
Subsampling Layer 1 (of 2) – **S2** • 2x2 kernel • Non overlapping	Keras AveragePooling2D Layer • pool_size= (2, 2) • strides= (2, 2)
Convolution layer 2 (of 3) – **C3** • 16 feature maps (28x28 in size) • 5x5 kernel	Keras Conv2D Layer • Filters=16 • Kernal size=5x5
Subsampling Layer 2 (of 2) – **S4** • 6 feature maps (5x5 in size) • 2x2 kernel	Keras AveragePooling2D Layer • pool_size= (2, 2) • strides= (2, 2)
Convolution layer 3 (of 3) – **C5** • 120 feature maps (1x1 in size)	Keras Dense Layer • units=120
Fully-Connected Layer – **F6** • 84 units	Keras Dense Layer • units=84
Output Layer – **OUTPUT** • 10	Keras Dense Layer • units=10
Layers C1 -> F6 • Sigmoid squashing function (hyperbolic tangent)	def custom_activation(x): return (K.tanh(2/3 * x) * 1.7159)

Learning rate scheduler
Global learning rate was scheduled as:
• 0.0005 for the first two passes
• 0.0002 for the next three
• 0.0001 for the next three
• 0.00005 for the next 4
• 0.00001 thereafter.
(20 training epochs in total)
	def lr_schedule(epoch, lr):
	if epoch < 2:
	return 0.0005
	elif epoch < 5:
	return 0.0002
	elif epoch < 8:
	return 0.0001
	elif epoch < 12:
	return 0.00005
	else:
	return 0.00001

Table 2 - Implemented functions.

2.4 Additional Experimentation on Replication Paper

In addition to the replication of the paper, we propose that we try and improve on the results of the paper (MNIST dataset) using different techniques. These are as follows:

2.4.1 Cross-validation (K-Fold)

This method splits the data into k folds, using [k-1] folds for training whilst holding back one of the folds for testing. The model is therefore trained k times with statistical analysis used to analyse the results (for example, accuracy). This is more computationally expensive as the training occurs k times, however, since it allows different data splits to be used, the results of which are averaged, it can give more balanced results as well as stopping data being 'wasted' on a validation split [13].

Key:	Training Data				
	Test Data				
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Figure 3 - k-fold cross validation with 5 folds (splits)

2.4.2 Hyperparameter tuning

Through experimentation with hyperparameters, it is proposed that we can validate if the 'best' parameters were chosen in the original experiment. For example, if we can achieve more accurate results from different hyperparameters it could be argued that incorrect values were chosen. If we find that the accuracy deteriorates as we vary the values, we can prove that the correct values were chosen.

2.5 Modifications and challenges

• Whilst it is relatively straightforward to match and model the original paper using Keras, the source code and software is undoubtably different to that used in the paper in 1998.

• The hardware used in the original experiment was a CPU on a "Silicon Graphics Origin server using a single 200 MHz processor" [1]. To be as faithful as possible to the paper where possible we trained our model using a CPU (as opposed to a GPU) but even so modern hardware is exponentially more powerful and architecturally more advanced. In the unlikely event that both hardware and software were available, we could assure our results more. Even with one of the above, it would be challenging to replicate due to likely compatibility issues. With no choice but to use modern tools and hardware this is a modification to the original experiment.

• The input image size required for LeNet-5 is 32x32 and the MNIST image size 28x28. This meant that padding had to be added equally so that the size matched.

• Hyperparameters will be tuned independently due to limited time and resources. They will also be limited to a small number of values as we do not have the resources to complete detailed experiments.

3. RESULTS

The original paper used the Error Rate, as a percentage, as the main performance metric. This was measured at regular intervals/on-the-fly at each training set iteration/epoch. The paper also describes when convergence happens and displays the results visually. All of these metrics can be achieved using Keras as our chosen tool and displayed visually using matplotlib.

Other metrics and visualizations such as a Confusion Matrix and Precision, Recall and F1 scores will be analysed in this paper.

3.1 Result on original dataset (MNIST)

3.1.1 Accuracy & loss

The mean accuracy and loss of the model trained on MNIST dataset for 20 epoch are visualised in Figure 4 and 5. The model was trained over 10 repetitions to reduce the variability that may arise due to randomness and to ensure the robustness of the model performance. As a result, the model was able to achieve a high classification accuracy on MNIST dataset.

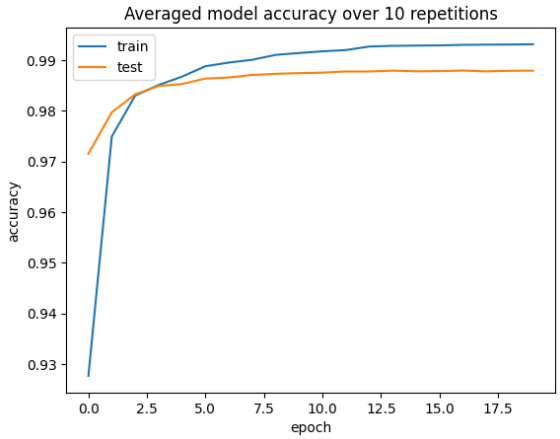

Figure 4 – The model's accuracy on training and test set of MNIST dataset in 20 epochs.

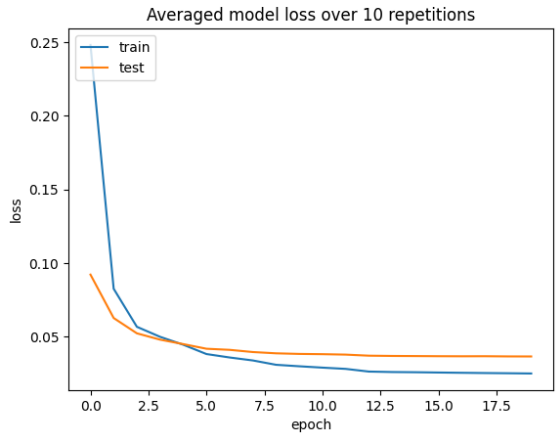

Figure 5 - The model loss on training and test set of MNIST dataset in 20 epochs.

3.1.2 Confusion matrix

The confusion matrix provides insights into the model's performance across different classes. Below is the confusion matrix for the replicated LeNet-5 model on the original dataset. Each cell in the confusion matrix represents the number of instances where a true label corresponds to the row and a predicted label corresponds to the column. High values along the diagonal indicate accurate predictions, while off-diagonal elements represent misclassifications.

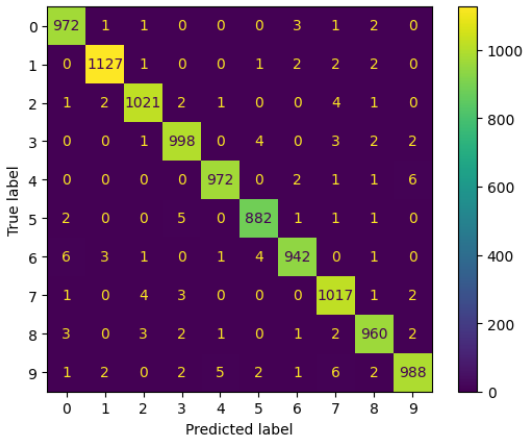

Figure 6 – The confusion matrix displaying model's performance on MNIST dataset.

3.1.3 Other metrics

In addition to accuracy and loss, the reproduced model performance was evaluated using other metrics including precision, recall and F1. The model achieved a precision of 98.77%, with a same value in recall and F1 score. Micro metrics are used in this model evaluation.

Metrics	Value
Loss	0.0507
Accuracy	98.88%
Precision	98.88%
Recall	98.88%
F1	98.88%

Table 3 – The table shows Loss, Accuracy, Precision, Recall, and F1 value upon the MNIST dataset.

3.1.4 T test

We conducted a one sample t test on the model accuracy. Confidence level is set at 95% with a degree of freedom at 9 for 10 repetitions. The purpose of performing this T test is to examine the accuracy of the model trained on MNIST with the one reported in the original paper to ensure that the result obtained from our model is not significantly difference. We set the null hypothesis (h0) as "there is no significant difference between accuracy from our model and the accuracy reported in the original paper" while the alternative hypothesis (h1) was set as "there is a significant difference between two results." Consequently, with our accuracy result, the null hypothesis was rejected, with a t-value of -9.592. (table…)

	Variable 1	Variable 2
Mean	0.98838	0.9905
Standard Deviation	0.00070	NA
Observations	10	1
Hypothesized Mean Difference	0	
Degree of Freedom	9	
t value	-9.592	
t Critical two-tail	2.262	

3.1.5 Cross Validation

The performance of the proposed model was also evaluated using k-fold cross validation, with the training and test datasets combined to form a single set, before being partitioned into five (k=5) equally sized subsets. The training process is repeated five times, with a different test dataset kept back each time leaving the model to train on the remaining 4 (Figure 3).

The validation accuracy across the five folds is shown in Table 5. The model achieved a mean loss of 0.0426 and a mean accuracy of 98.72%, with a standard deviation of 0.0012. The variation in the results shows that using this method helps reduce the overall variance (though averaging techniques), avoid overfitting (as the model is 'shown' different subsets of the data each time) as well as making the nest use of the data.

K	Validation Accuracy
1	0.987314 1
2	0.989392 2
3	0.985808 3
4	0.986884 4
5	0.986525 5

Table 4 – Accuracy in 5 folds

The model loss in cross validation training process was also plotted, with fold 2 having the lowest test loss (Figure 7).

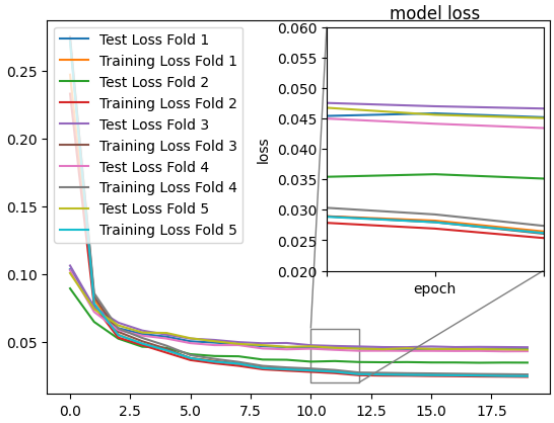

Figure 7 – The model loss on training and test set of 5 folds

3.2 Hyper parameter tuning

Hyperparameter tuning was performed to optimize the performance of the model on the Fashion MNIST (FMNIST) and MNIST datasets. The tuning process involved varying the kernel size, learning rate multiplier, and optimizer configuration.

3.2.1 Setup

Table 5 – The rejection of null hypothesis regarding the result

The dataset was split into training, validation, and test sets in a ratio of 54000:6000:10000 samples. The hyperparameter tuning experiments were conducted on the validation set, while the test set was used to evaluate the final performance of the tuned model.

3.2.2 Kernel size

Three different kernel sizes (3, 5, and 7) were evaluated to determine their impact on model performance. The model loss

over 20 epochs, using these three kernel sizes is shown as below (Figure 8).

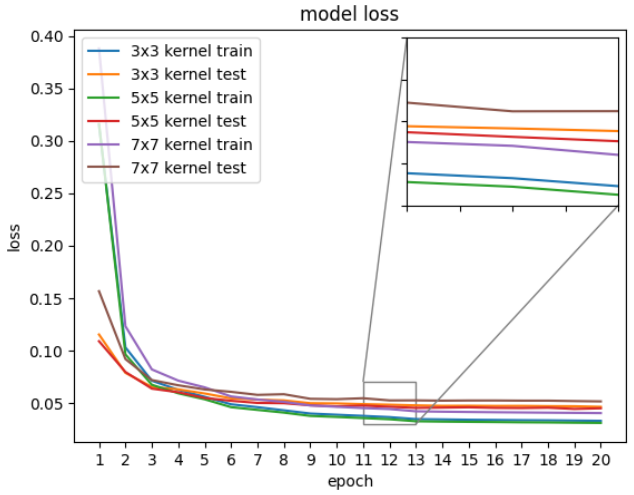

Figure 8 – Model loss of training and test set on different kernel sizes

3.2.3 Learning Rate

Different multipliers on the original learning rate schedule were tested to scale the learning rate during optimization. For the FMNIST dataset, the multipliers 1, 5, 10, 15, 25, 50, and 100 were considered, while for the MNIST dataset, the multipliers 1, 5, 10, 50, and 100 were evaluated.

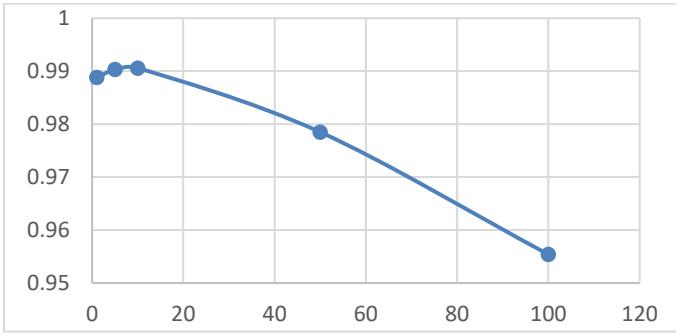

Figure 9 - MNIST model accuracy vs learning rate.

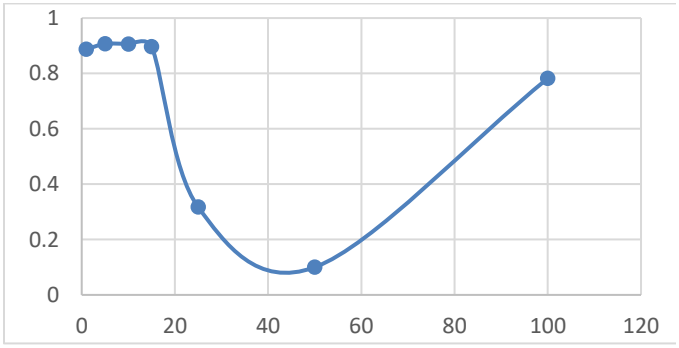

Figure 10 - FMNIST model accuracy vs learning rate.

An additional one-sample t-test was conducted to compare the accuracy obtained with a learning rate multiplier set to 10 against the result reported in the original paper. The null hypothesis assumed that there was no significant difference between the mean accuracy obtained in our experiment and the accuracy reported in the original paper was accepted, leading to the conclusion that our model performed the same level with the one used in 1989.

	Variable 1	Variable 2
Mean	0.99045	0.9905
Variance	0.00075	NA
Observations	10	1
Hypothesized Mean Difference	0	
Degree of Freedom	9	
t value	-0.212	
t Critical two-tail	2.262	

Table 6 – The acceptance of null hypothesis

3.2.4 Optimizer

Hyperparameter tuning experiments were conducted to compare the performance of different optimisers, including standard ADAM (ADAMhistory), ADAM with weight decay (ADAMWhistory), and RMSprop (RMShistory). A line plot in figure 11 was generated to visualize the error trends of models trained with different optimisers over the course of training epochs.

3.3 Results on Fashion-MNIST Dataset

3.3.1 Accuracy & loss

The model was trained on Fashion-MNIST to test its performance on modern datasets. The model loss over 20 epoch was plotted as below.

3.3.2 Confusion matrix

The confusion matrix for the Fashion MNIST model is presented below, with the corresponding labels shown in the table.

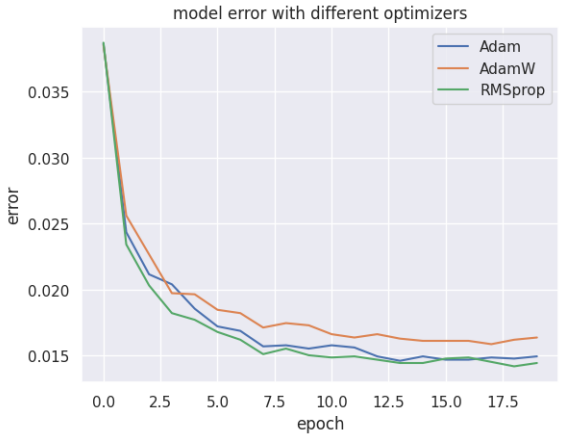

Figure 11 – Model error with different optimisers.

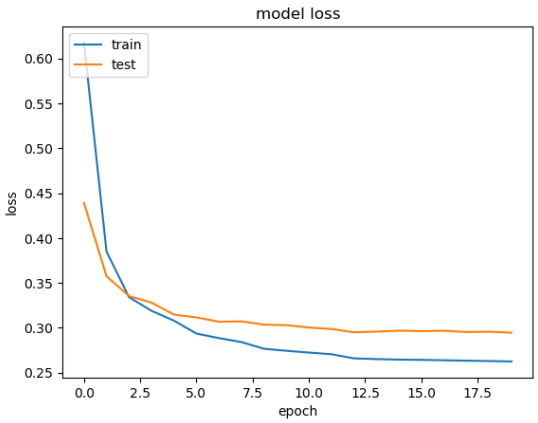

Figure 12 - The model loss on training and test set of F-MNIST dataset in 20 epochs.

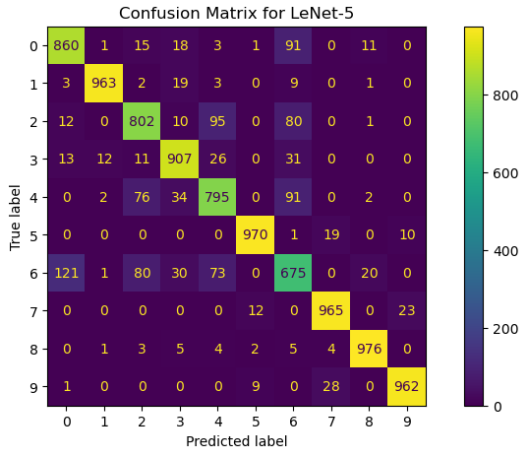

Figure 13 - Confusion matrix displaying model's performance on FMNIST dataset.

	Label		Label
0	T-shirt/top	**5**	Sandal
1	Trouser	**6**	Shirt
2	Pullover	**7**	Sneaker
3	Dress	**8**	Bag
4	Coat	**9**	Ankle boot

Table 7

3.3.3 Other metrics

The model achieved an accuracy of 88.75% and a loss of 0.3093. Other metrics are shown as below. Five significant figures were included to show the metrics difference. Macro metrics are used in this dataset evaluation.

Metrics	Value
Loss	0.2995
Accuracy	89.100%
Precision	89.067%
Recall	89.100%
F1	89.073%

Table 8 – Model Loss, Accuracy, Precision. Recall, and F1 value over FMNIST dataset.

3.4 Results on Extended-MNIST Dataset

3.4.1 Accuracy & loss

The model was trained on Extended-MNIST as well to test its performance on modern datasets. The model loss over 20 epoch was plotted as below.

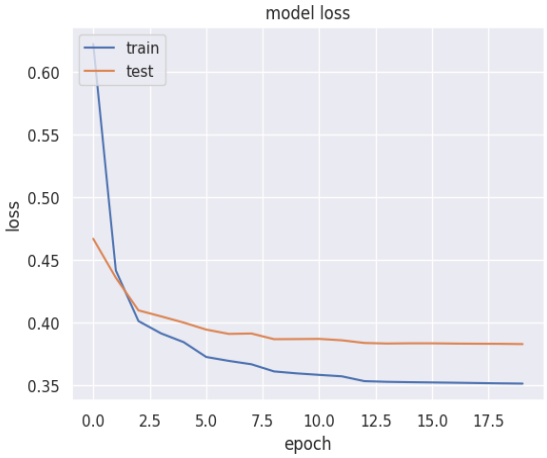

Figure 14 – Model loss over training and test set of EMNIST dataset in 20 epochs.

3.4.2 Confusion matrix

The confusion matrix for the Fashion MNIST model is presented below, with the corresponding labels shown in the table.

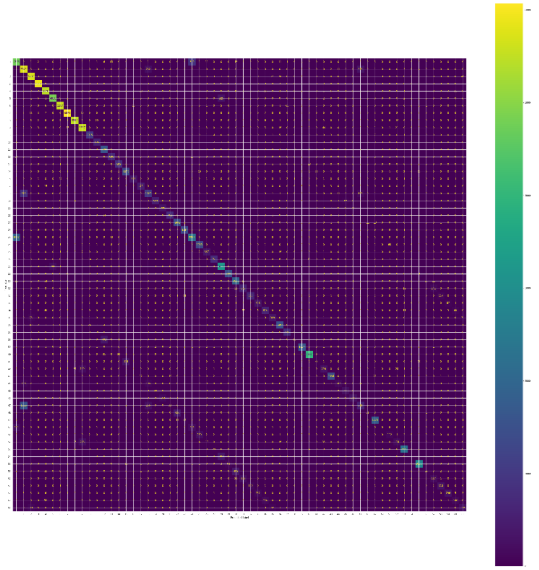

Figure 15 – Confusion matrix displaying model's performance on EMNIST dataset.

3.4.3 Other metrics

The model achieved an accuracy of 86.32% and a loss of 0.3768. Other metrics are shown below. Micro metrics are used in this case.

Metrics	Value
Loss	0.3768
Accuracy	86.32%
Precision	86.32%
Recall	86.32%

F1	86.32%

Figure 16 – Model Loss, Accuracy, Precision. Recall, and F1 value over EMNIST dataset.

4. Discussion

4.1 Interpretation of Results

The Lecun et al. [1] paper declared an accuracy result of 99.05%. During our tests, we were achieved 98.83% accuracy over 10 observations, therefore we rejected the null hypothesis (H0) proposed by the one sample t-test. From our experiments we can conclude the original paper cannot be replicated with the information provided to the same levels of accuracy at the R2 (Data Reproducibility) level.

It was therefore interesting to note that with hyperparameter experimentation it is possible to meet the threshold for accepting the null hypothesis.

There are multiple reasons this could be the case. For example, we lack the concrete implementation used in the paper so could be missing some of the critical components needed to replicate the study and the lack of 'Experiment' data (as per Semmelrock et al.) [3] and/or the stochastic nature of model training. It is notable that concrete versions of deep learning frameworks have a big impact on the results [21].

When comparing the original paper error rate results to the replicated results we can see a very similar pattern with the biggest falls in the error rate between the first few epochs, seeing stabilisation and convergence occurring earlier in our experiment. One explanation for this could be the efficiency of modern tools, though again more experimentation is needed to prove this.

Another pattern that has been replicated by our experiment is the training error rate starting higher than the test error rate, then converging roughly 0.5% below the test error rate. The explanation given in the paper for this is "the average training error is measured on the fly as training proceeds;" given we experienced the same results it would suggest this could be true for our experiment.

We proceeded to explore a few misclassified results and found the images hard to classify with the naked eye. From a human perspective, this is due to variability of handwriting but this could mean there are mistakes in the data. Whilst still possible to improve performances, it would need to be completed carefully as to increase accuracy without overfitting on these 'mistakes'. This is backed up by Cohen et al. suggesting at very high classification accuracy "the dataset labelling can be called into question." [10] The confusion matrix for the MNIST experiment confirms this potential issue with 7-1, 5-3 and 4-1 the largest categories incorrectly classified.

4.2 Insights from Modern Datasets

The aim of the EMNIST dataset is to create a more challenging classification task that involves letters as well as digits, in the same format used for the MNIST dataset [10]. Since the dataset is in the same format it enables is to provide a quick, quantitative and fair means of comparing approaches [10].

The results from training the LeNet-5 replication using the EMNIST dataset (86.32%) were not as good as the MNIST (99.03%). The EMNIST algorithm has four times as much data and 62 classes (vs. 10 in MNIST) and we would therefore expect the performance to be lower. Having more classes means an increased margin for error where images are misclassified with many combinations of similar shaped classes present (for example, I, 1, 7, T).

Similarly, when we trained a model on FMNIST using the replicated LeNet-5 environment the accuracy score was 88.75%. This is much lower than the original MNIST but above that of EMNIST. From the raw data it seems that the overall performance is better on FMINST vs. EMNIST, however, this comparison is not as straightforward as there are far fewer classes (10) in the FMIST datasets when compared to the EMNIST (62), leading to a greater risk of misclassification.

According to the FMINST authors accuracy table, (untested) classification accuracy is as 96.7% and the average of those (untested) results listed is 92.36%; 6.04% higher than our results (10.37% difference vs. the highest scoring) which shows significant room for improvement.

The LeNet-5 network though state-of-the-art at the time, is relatively small when compared to today's architectures. Increasing the size or number of hidden layers may give better performance; this was seen by Cohen et. al as they tested their network on the EMNIST dataset [10]. This can also be seen as early as 2011 where Ciresan et al. reduced the error rate to 0.27% [19].

4.3 Comparative Analysis

Both the EMNIST and FMINST datasets are designed as 'drop in replacements' for the original dataset. This allows for a much fairer experiment and makes comparison much more straightforward.

The validation and test losses did not noticeably diverge in any of our experiments, which could otherwise indicate overfitting [20]. The original paper notes that trend does not occur and suggests this is down to the leaning rate being kept relatively large [1]. From our initial results experimentation with the EMNIST and FMINST datasets, this is also observed.

The original LeNet-5 architecture [1] was designed for a specific task; handwriting recognition. It is likely that this would not directly transfer to other domains without modifications to the network.

The LeNet-5 network performs reasonably well on the EMNIST dataset, but as it was originally designed for the 10 classes used in MNIST it is likely not optimised for this task.

Similarly, FMNIST contains 10 classes but overall the images provided are much more complicated than the MNIST and EMNIST datasets, meaning the network is not likely to be optimal. For example, the shallow LeNet-5 network may struggle with feature extraction and detecting the (additional) edges found in the images.

4.4 Limitations

There are several limitations in our study. These are described below:

- There is no source code making it difficult to replicate the paper at R1 level [3]
- The software used in the original experiment is not listed, meaning there may be differences between the techniques used when compared to more modern tools (e.g. Keras)
- There may be differences between the architecture of CPU's used in the experiment compared to that found in up-to-date hardware

- The dataset(s) used are relatively clean; this aids reproducibility but may hinder any real-world use of the model as production data is not typically this precise
- Our experiment did not compare the CNN architecture with the classifiers used for comparison in the original paper [1] which would have allowed for an additional replication comparison to be made
- For time reasons, no additional imagery pre-processing experimentation took place.
- Our hyperparameters were trained independently to one another and also manually. This was due to a lack of time and compute.
- Our study only explored k-fold techniques; k-fold is only one of a number of other cross validation strategies. These could also be explored.
- Our study did not test larger image sizes. It is hypothesised that this would have posed a challenge for the architecture given the shallow depth. This would also give some insight into real-world application of the model.
- Our study did not look at how the characters are segmented from documents. As with larger image sizes, this would provide a more authentic and holistic use case.
- Our study (and the paper) only tested accuracy, not efficiency. We cannot be sure that the original papers methods performance can be improved without detailed further study.

5. Conclusion

5.1 Summary of Key Findings

We were able to achieve to 98.83% accuracy vs. the 99.05% in the original paper [1]. We conducted a one sample t-test where the t-value fell outside of the range of the critical values. This means we reject the null hypothesis (H0) proposed; that "the original mean equals the reproduced mean value." We conclude the original paper cannot be replicated from the information provided.

The detailed experiment notes in the original paper undoubtedly helped us reach our experiments accuracy level, however, it was not enough to statistically confirm replication. The explanation of the network, hyperparameters, the test/train split used and other topology details such as the learning rate and activation function all facilitated reproduction.

Our experiment shows Cross Validation techniques are very important and are likely to improve the overall accuracy of the result. The k-fold validation performed shows that the loss/accuracy varied by ~2.5%. Taking an average of the folds helps reduce the impact of potential problems such as an imbalanced test/train split.

Hyperparameter tuning was also proven to have a substantial impact on the model training, as demonstrated in our results. It was found that some of the hyperparameters we chose to experiment with could improve results, whereas others were already optimal in the original paper.

Some of the misclassifications by our model (and the original paper [1]) are very difficult to determine for a human. This isn't necessarily a problem for a machine given the data is valid, as proven multiple times in cancer diagnosis where ML models frequently outperform human detection rates [22]. If, however, the data is indeterminable it may impact model training.

5.2 Reflections on the Process

Insights gained can broadly be split into two categories:

1) Insights gained from the replication exercise

Replicating a paper showed the critical importance of detailing the exact steps used when running an experiment. The significant improvements shown in neural network capability in the paper is completely independent of how the researchers chose to document their findings, but the level of detail provided allows a level of replication that would have an impact on anyone that wanted to progress the proposed ideas further. Having the data readily available is another example of how our experiment and countless research experiments since have been able to progress the state-of-the-art and compare results to validate improvements.

In our experience should details such the original code/software packages have been provided we may have been able to get the experiment up and running much quicker and replicate it more accurately (with the only difference being the hardware). From the limited research and reading of other papers as a part of this experiment, I would also conclude the same as Semmelrock et al. [3] that replicability is an issue due to a lack of experiment details.

2) Insights gained from replicating the paper

As described above, the replication showed us the impact that hyperparameter tuning and cross-validation strategies can have. Changing these hyperparameters allowed us to improve on the transparency of the original paper by showing if the choices made were optimal.

We are fortunate that the original dataset is still available, which provides us with the test/train spit. Without this, the cross-validation strategy and any repetition and averaging of the experiment become more important as it is very unlikely that the exact same test/train split would be achieved therefore making replication harder. Similarly, the datasets we used were 'out of the box' and ready to use with little to no preparation required. If the MNIST dataset wasn't available or if it was 'dirty', we would require a cleansing step. This could add significant variability to any experiment unless the exact cleansing steps used were listed within the paper.

The replicated experiment required a large amount of labelled data – this is often one of the hardest things to obtain when training models as it is either cost or resource intensive [23]. This would add complexity to carrying out further experimentation, which may not be the same for unsupervised model training techniques [24]. In contrast, supervised techniques such as LeNet-5 may avoid a lack of transparency and allow for higher accuracy.

5.3 Future Work

There are multiple areas in which the research we have started could continue and have value in doing so. These are listed by category below:

Improving the replication experiment

To validate and further prove the replication, different Machine Learning tools could be used. For example, sk learn or cloud based tools (Azure, AWS) to see if the results differ significantly. Testing across CPU's and then comparing these results with GPU's would provide a means of assessing the impact hardware has on the results.

Extending the replication experiment

Preprocessing the data in different ways could be used to try and increase performance. Similarly, using larger images and data related to what might be seen in a productionised environment could be used to train the model and for inference.

Hyperparameter grid search should be used for future experiments to take advantage of automation. Different values for k in the k-folds cross validation should also be trialled to assess any impact. Different cross-validation strategies should be explored; this is true even within the k-fold technique where stratified and repeated K-fold techniques could be utilised [25]. State-of-the-art hyperparameters techniques could be trialled, including the Genetic Algorithm (GA) and Evolution Strategies (ES) [16]

The original papers network was constrained by technology at the time – it would be interesting to explore the impact creating a larger network would have on the results. Cohent et al. describe the accuracy improving as the size of the hidden layer(s) increases.

Given the costs associated with creating datasets such as MNIST, EMNIST and FMNIST, it could be viable to explore the creation of synthetic data for training future models.

Additional dataset experimentation

It was not possible to repeat some of the tests (such as cross validation and experiment repetition and averaging) on the EMNIST and FMNIST datasets due to a lack of time and compute resource. With an appropriate budget, using cloud based architecture could help overcome this limitation.

Testing other Machine Learning techniques

Training using other techniques could provide evidence as to how relevant the model is. This could be done retrospectively using those in the original paper or using more modern techniques. For example, how do contemporary Natural Language Processing (NLP) and Large Language Model (LLM) tools perform in comparison? Model ensembles have not been considered and could be a valuable area of research (performing a literature review to see what work has already been done).

In these experiments efficiency could be a key metric to consider.

Implementing tools to aid experimentation and replication

There are many ways to improve the data scientist's efficiency and therefore effectiveness, as well as the experiment efficiency. For example, there are open source [26] and hyperscale-cloud provided tools [27] that make use of MLOps frameworks. This type of environments significantly reduces the manual process required when training models, including no-code solutions. They can help keep track of results and experiments though repeatable training workflows, cataloguing artifacts (including versions) and offering solutions for deploying and maintaining models.

Containerisation should be strongly considered for any future replication experiments. The environment can be easily standardised within the container meaning the choice of operating system becomes less of an issue. Should scaling out need to be considered, containerisation makes the process much easier [28]. Lastly, but potentially most importantly, containers offer a way of providing near instant replication on any environment that supports tools such as Docker, since the experiment can be packaged and listed on sites such as Docker Hub [29].

REFERENCES

[1] Yann LeCun, Leon Bottou, Yoshua Bengio and Patrick Haner, 1998. Gradient-Based Learning Applied to Document Recognition. DOI: 10.1109/5.726791 Proceedings of the IEEE, Volume 86, Issue 11, Nov 1998. Page 13-16, Fig. 9. and Fig.12.

[2] Yann LeCun, Leon Bottou, Yoshua Bengio and Patrick Haner, 1998. Gradient-Based Learning Applied to Document Recognition. DOI: 10.1109/5.726791 Proceedings of the IEEE, Volume 86, Issue 11, Nov 1998. Page 6-7, Paragraph A: *Convolutional Networks*.

[3] Semmelrock, H., Kopeinik, S., Theiler, D., Ross-Hellauer, T., Kowald, D., 2023. Reproducibility in Machine Learning-Driven Research. Page 7.

[4] Gundersen, O.E., Kjensmo, S.: State of the Art: Reproducibility in Artificial Intelligence. Proceedings of the AAAI Conference on Artificial Intelligence 32(1) (2018). https://doi.org/10.1609/aaai.v32i1.11503, https://ojs.aaai.org/ Index.php/AAAI/article/view/11503.

[5] Red Hat Developer. (2022). "Explore the MNIST dataset." [Online]. Available: https://developers.redhat.com/learning/learn:openshift-ai:how-create-tensorflow-model/resource/resources:explore-mnist-dataset. [Accessed: 12-Feb-2024].

[6] TensorFlow "mnist | TensorFlow Datasets" [Online]. Available: https://www.tensorflow.org/datasets/catalog/mnist. [Accessed: 12-Feb-2024].

[7] PyTorch.org. "MNIST — Torchvision main documentation" [Online]. Available: https://pytorch.org/vision/main/generated/torchvision.datasets.MNIST.html. [Accessed: 12-Feb-2024].

[8] Yann LeCun, Corinna Cortes, Christopher J.C. Burges " THE MNIST DATABASE of handwritten digits " [Online]. Available: http://yann.lecun.com/exdb/mnist/index.html. [Accessed: 12-Feb-2024].

[9] Zalando Research Github "fashion-mnist" [Online]. Available: https://github.com/zalandoresearch/fashion-mnist. [Accessed: 12-Feb-2024].

[10] Gregory Cohen, Saeed Afshar, Jonathan Tapson and André van Schaik, 2017. G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," in Proc. IEEE Int. Joint Conf. Neural Netw., 2017, pp. 2921–2926.

[11] G. Cohen, S. Afshar, J. Tapson and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 2017, pp. 2921-2926, doi: 10.1109/IJCNN.2017.7966217.

[12] Keras "Keras Documentation" [Online]. Available: https://keras.io/ [Accessed: 12-Feb-2024].

[13] SciKit-Learn "3.1. Cross-validation: evaluating estimator performanc " [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html [Accessed: 23-Apr-2024].

[14] Sannella, M. J. 1994. *Constraint Satisfaction and Debugging for Interactive User Interfaces*. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-09398., University of Washington.

[15] Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.* 3 (Mar. 2003), 1289-1305.

[16] Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology* (Vancouver, Canada, November 02 - 05, 2003). UIST '03. ACM, New York, NY, 1-10. DOI= http://doi.acm.org/10.1145/964696.964697.

[17] Yu, Y. T. and Lau, M. F. 2006. A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions. *J. Syst. Softw.* 79, 5 (May. 2006), 577-590. DOI= http://dx.doi.org/10.1016/j.jss.2005.05.030.

[18] Spector, A. Z. 1989. Achieving application requirements. In *Distributed Systems*, S. Mullender, Ed. ACM Press Frontier Series. ACM, New York, NY, 19-33. DOI= http://doi.acm.org/10.1145/90417.90738.

[19] D. C. Ciresan, U. Meier, L. M. Gambardella and J. Schmidhuber, "Convolutional Neural Network Committees for Handwritten Character Classification," *2011 International Conference on Document Analysis and Recognition*, Beijing, China, 2011, pp. 1135-1139, doi: 10.1109/ICDAR.2011.229.

[20] AWS " What is Overfitting?" [Online]. Available: https://aws.amazon.com/what-is/overfitting/. [Accessed: 24-Apr-2024].

[21] Shahriari, Mostafa, Rudolf Ramler, and Lukas Fischer. 2022. "How Do Deep-Learning Framework Versions Affect the Reproducibility of Neural Network Models?" Machine Learning and Knowledge Extraction 4, no. 4: 888-911. https://doi.org/10.3390/make4040045

[22] Jong Hyuk Lee, Hyunsook Hong, Gunhee Nam, Eui Jin Hwang, Chang Min Park, 2023. "Effect of Human-AI Interaction on Detection of Malignant Lung Nodules on Chest Radiographs" *Radiology* https://doi.org/10.1148/radiol.222976

[23] Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want To Reduce Labeling Cost? GPT-3 Can Help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205, Punta Cana, Dominican Republic. Association for Computational Linguistics.

[24] IBM " Supervised vs. Unsupervised Learning: What's the Difference?" [Online]. Available: https://www.ibm.com/blog/supervised-vs-unsupervised-learning/. [Accessed: 24-Apr-2024].

[25] Scikit Learn "Cross-validation: evaluating estimator performance" [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html. [Accessed: 24-Apr-2024].

[26] RedHat "Open Source MLOps with ZenML and OpenShift" [Online]. Available:

https://www.redhat.com/en/blog/open-source-mlops-with-zenml-and-openshift. [Accessed: 24-Apr-2024

[27] AWS "Amazon SageMaker Studio" [Online]. Available: https://aws.amazon.com/sagemaker/studio/. [Accessed: 24-Apr-2024

[28] AWS "AWS Deep Learning Containers" [Online]. Available: https://aws.amazon.com/machine-learning/containers/. [Accessed: 24-Apr-2024

[29] Docker "DockerHub" [Online]. Available: https://hub.docker.com/. [Accessed: 24-Apr-2024