Lightweight Single Model CNN (SveltNet CNN)

Nikolaos Kylintireas
Electrical and Computer Engineering
Aristotle University of Thessaloniki
Thessaloniki, Greece
nickilintireas@gmail.com

Evangelos Lourmpakis
Electrical and Computer Engineering
Democritus University of Thrace
Xanthi, Greece
evanlourb@gmail.com

Christodouli Toramanidou
Electrical and Computer Engineering
Aristotle University of Thessaloniki
Thessaloniki, Greece
chtoraman@gmail.com

***Abstract*—This paper features a lightweight neural network designed for efficiency and broad applicability in classification tasks. Initially evaluated on the MNIST dataset, the model achieves competitive performance utilizing significantly fewer parameters than many current state-of-the-art models. Its architecture offers a promising solution for deployment in resource-constrained environments and general-purpose applications.**

***Keywords*—Convolutional Neural Network, Deep Learning, Machine learning, MNIST Dataset, TensorFlow**

I. INTRODUCTION

Deep learning models have achieved remarkable success across various domains; however, many widely used architectures remain prohibitively complex for resource-constrained environments. This paper introduces a lightweight neural network that maintains competitive performance while significantly reducing computational overhead. Initially evaluated on the MNIST dataset [1], an established benchmark for handwritten digit recognition, the model was later tested on the Fashion-MNIST [2] and CIFAR-10 [3] datasets. The proposed model not only achieves accuracy comparable to more complex systems but also utilizes significantly fewer parameters. Moreover, its streamlined design enhances adaptability, making it applicable to a wide range of tasks beyond digit recognition. This versatility, combined with its efficiency, positions the model as a promising solution for both specialized and general-purpose applications in machine learning.

II. DATA PREPARATION

A. Image Preprocessing

Image preprocessing is a widely used technique in machine learning to improve model performance and adaptability. However, it was not included in this project, as the images in the datasets were relatively straightforward or even preprocessed. Furthermore, testing results proved that incorporating preprocessing could limit the model's versatility, as it currently demonstrates strong performance across a wide range of applications.

B. Data Augmentation

The training data was augmented using the ImageDataGenerator function, imported from the tensorflow [4] library. The augmentation techniques applied include:

- Rotation (±20 degrees): Randomly rotates images within a 20-degree range to introduce variance in orientation.
- Width and Height Shifts (±15%): Translates images horizontally and vertically by up to 15% of the image size to improve spatial invariance.
- Shear Transformation (±20%): Applies affine transformations to slightly tilt the images
- Zoom (±20%): Randomly scales images within a 20% range to make the model more resilient to variations in object size.

Each of the aforementioned values could be modified to extract better performance from any specific dataset, depending on its properties.

III. MODEL ARCHITECTURE

As referenced above, the main objective of the model is to work accurately enough with the least amount of resources possible. To accomplish this, careful consideration must be given to the architectural design. Since the introduction of the groundbreaking AlexNet model in 2012 [5][6], which popularized several now-standard techniques such as ReLU activation functions, data augmentation, dropout layers, and overlapping pooling, most subsequent models have built upon its foundation. While these advancements have led to improved accuracy, they have largely done so by increasing the number of layers and parameters.

A. SveltNet's principals

The main principals of the model's architecture are simplicity and countering overfitting. Other models widely used today featuring simple solutions are the Xception [7][8] model, which manages to match much more complex models while keeping a relatively low amount of resources on hand and the Squeezenet [9] architecture, which manages to incorporate AlexNet levels of accuracy into a much more compact size. These examples are proof that a model does not need extensive depth and parameter amount in order to be effective when used for common classification problems.

The over-reliance on training data in order to recognize a sample picture, often reffered to as overfitting, leads to significant losses when trying to classify meaningful data. . This happens when the training data is insufficient or the model is over-reliant to its input.. The solution to this problem is not one-dimensional. Dropping the effective layers of the model also usually drops the general accuracy and methods such as dropout, kernel regularization and data augmentation do not manage to fully nullify this phenomenon. In general, a model needs to have both sufficient depth and a limited reliance on training data.

B. Assessing depth-wise separable convolutions

An optimal solution may lie in Depth-wise separable convolutions [10][11]. Generally, they help keeping both the parameter number low and limit the over-reliance on previous layer data. They are generally used in limiting the amount of processing needed internally when working with multi-channel data, such as colored images. As shown in the testing results section, when used in combination with a smaller amount of traditional convolutional layers along with dropout, they can be used as a cheap parameter-wise substitute for extending the depth of our model without triggering over-fitting, even when used with single channel data.

C. Model Design

The SveltNet model has 3 main parts:

- The top part, which consists of the input layer for each subsequent dataset, followed by 2 depth-wise convolution layers.
- The main part, which consists of 4 sets of a traditional convolutional layer followed by down-sampling the data, a dropout layer and passing the data to two depth-wise convolutional layers.
- the output, which consists of global average pooling, dropout and finally using a dense layer to finalize the output. This ensures minimal over-fitting, even on much smaller datasets.

Fig. 1 shows a complete diagram of SveltNet's architecture, featuring all of the above.

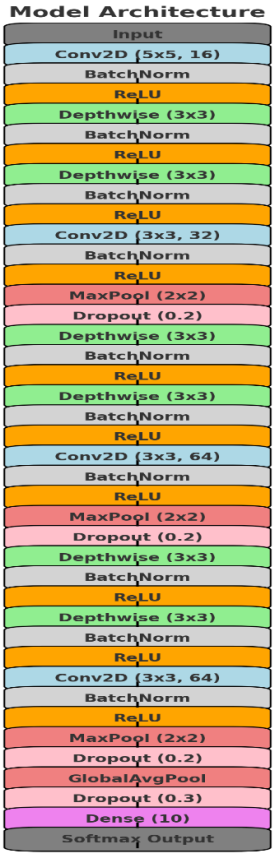

Figure 1: SveltNet architecture

IV. Testing Results and Evaluation

The SveltNet model was evaluated using a 20% train-validation split for each dataset. The Adam optimizer from the TensorFlow library was utilized to optimize performance, while accuracy, precision, and recall were chosen as evaluation metrics. Additionally, the loss function was customized with label smoothing [12][13] to mitigate overconfidence in predictions and improve generalization. The final summary of trainable and non trainable parameters is around 100.000.

A. MNIST results

Fig. 1 features the results of the model after training based on the aforementioned information. A total of 50 epochs were used for the training, seemingly enough to reach the accuracy plateau. This figure as much as the following showcase an underfitting behavior, which is to be expected given the generality of the model.

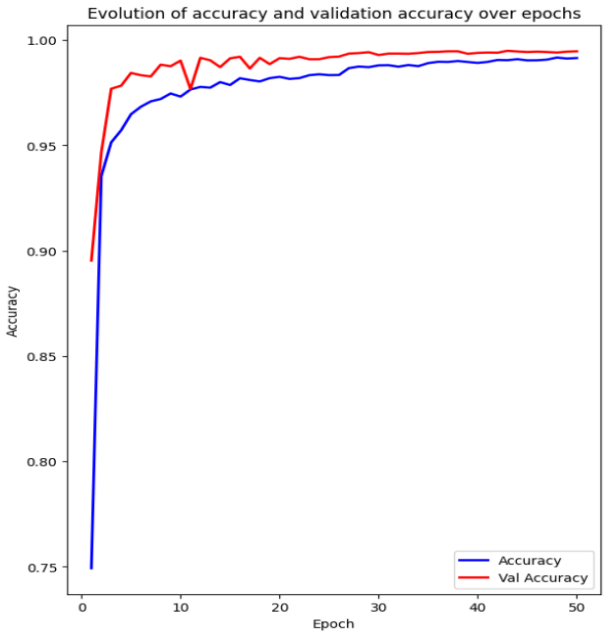

Figure 2: Accuracy vs Epochs for the MNIST dataset

B. Fashion-MNIST results

Similarly, Fig. 2 presents the results obtained on the Fashion-MNIST dataset. In this case, the learning curve exhibited a less steep progression, indicating a slower convergence. To address this, the number of training epochs was increased to 60 to ensure sufficient learning. This behavior was anticipated, as Fashion-MNIST is inherently more complex than MNIST [2].

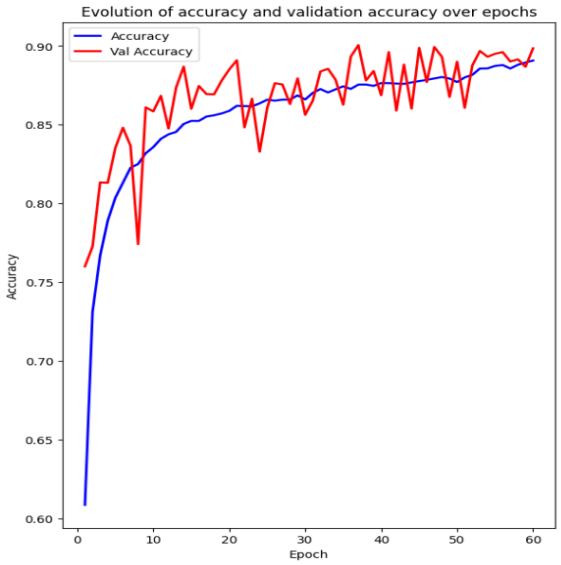

Figure 3: Accuracy vs Epochs for the Fashion-MNIST dataset

An interesting observation is that the validation accuracy appears significantly less stable during the first 50 epochs

compared to Fig. 1. However, it stabilizes toward the end, ultimately reaching the expected plateau and exhibiting slight underfitting behavior.

C. CIFAR-10

Finally, Fig. 3 illustrates the model's performance on the CIFAR-10 dataset. Among the three datasets, this resulted in the least favorable learning curve. To address this, the number of training epochs was increased to 80 for the subsequent analysis.

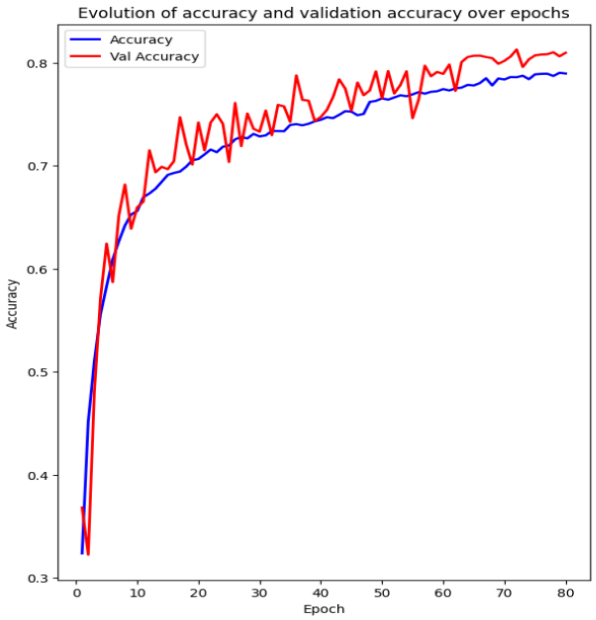

Figure 4: Accuracy vs Epochs for the CIFAR-10 dataset

The underfitting observed in the figures above can be addressed by adjusting the dropout values for each specific dataset, which could further enhance accuracy without compromising performance. However, these adjustments are highly dataset-dependent and should not contribute to the general-purpose nature of the current model.

D. Comparison between AlexNet, SqueezeNet, SveltNet

Fig. 4 contains a direct comparison between AlexNet, SqueezeNet and this paper's featured model, SveltNet in the three aforementioned datasets. It should be noted that SqueezeNet results include two different models: one[1] for the Fashion-MNIST dataset and a different one[2] for CIFAR-10.

FIGURE 4: TABLE I

Datasets	**Models**			
	Squeezenet (761k)	**Squeezenet** (120k)	**Alexnet** (57M)	**SveltNet** (100k)
MNIST	-	-	98.71%	99.63%
Fashion-MNIST	85%	-	92%	89.45%
CIFAR-10	-	80%	83.47%	79.93%

V. Usage and Example UI

To test the model and evaluate its performance in a real-time scenario, a simple stand-alone application was developed to test the digit recognition abilities and to ensure the response time of such application remains acceptable for this lightweight model. The application allows users to either draw a digit on a canvas or upload an image for prediction. It includes functionality for digit recognition and features a "Clear" button to reset the canvas or remove the uploaded image. After each prediction, the corresponding inference time is displayed at the bottom. The user interface was implemented using the PyQt5 library [14].

A. Image Input

When an image is captured from the canvas, the QPixmap object containing the drawing is converted into a QImage using the .toImage() method. The resulting image is in a 32-bit RGBA format, where each pixel consists of four channels: red, green, blue, and alpha (transparency). Similarly, when an image is uploaded using the "Upload" method, it is initially in RGB format and must be converted to grayscale for consistency with the training data. It is crucial that both the canvas-drawn and uploaded images are resized to approximately 28×28 pixels to align with the MNIST dataset's input dimensions.

B. Image Processing

To ensure that captured images serve as valid input for the model, they must first be converted to grayscale while preserving contrast and transforming them into a black-and-white format. This is achieved using QImage.Format_Grayscale8. The pixel data is then stored as a flat byte array and reshaped accordingly. Initially, the array is reshaped to match the canvas size (280×280) before being resized to 28×28 using interpolation to align with the MNIST model's input dimensions.

A critical step in this process is Otsu's thresholding [15], which binarizes the image based on pixel intensity. The purpose of this method is to determine the optimal black-and-white values for each pixel, thereby converting the image into a true binary representation. cv.THRESH_OTSU automatically identifies the ideal threshold by analyzing the image histogram and maximizing the between-class variance. Additionally, cv.THRESH_BINARY_INV is applied to invert the binary thresholding, ensuring that the digit appears white against a black background. This step is crucial for maintaining consistency with the test images in the dataset.

Finally, pixel values are normalized by scaling them to [0,1] through division by 255. Once these preprocessing steps are completed, the image is ready to be passed into the predict method for classification.

C. Real case scenarios and scalability

This digit recognition model can be expanded further into a vast variety of applications. Firstly it could be an important step towards the digitization and automation in industries. With further development such as text recognition, it can be incorporated in systems where numerical handwritten data need to be transformed into digital data, such as data extraction via scanning (OCR), product management and tracking, grading systems in education, reports etc.

By leveraging the OpenCV library, more advanced applications can be developed. This could be proven useful in

pattern and image recognition in AI-Powered assistive technology, varying from voice-over-text assistance to autonomous driving. Due to its lightness and high accuracy, the model is promising towards security, easy application and customization solutions.

REFERENCES

[1] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, *29*(6), 141–142

[2] Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Han Xiao, Kashif Rasul, Roland Vollgraf.

[3] Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo,Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis,Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow,Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia,Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster,Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens,Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker,Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas,Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke,Yuan Yu, and Xiaoqiang Zheng.TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[5] A. Krizhevsky, I. Sutskever, G. E. Hilton: ImageNet Classification with Deep Convolutional Neural Networks.

[6] N. Klingler: AlexNet: A Revolutionary Deep Learning Architecture.

[7] F. Chollet: Xception: Deep Learning with Depth-wise Separable Convolutions.

[8] G. Boesch: Xception Model: Analyzing Depthwise Separable Convolutions

[9] SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size

Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer

[10] S L. Kaiser , F. Chollet, A. N. Gomez: DEPTHWISE SEPARABLE CONVOLUTIONS FOR NEURAL MACHINE TRANSLATION

[11] “Y. Guo, Y. Li, L. Wang, T. Rosing: Depthwise Convolution Is All You Need for Learning Multiple Visual Domains

[12] Mezzini, M. (2018). Empirical study on label smoothing in neural networks. In Proceedings of the 26th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (pp. 25-32).

[13] TensorFlow Keras API Documentation. CategoricalCrossentropy

[14] Riverbank Computing Limited. (2024). PyQt5 (Version 5.15.11) [Python package]. Python Package Index.

[15] Srinivas, A. (2020, December 22). *Otsu thresholding with OpenCV*. LearnOpenCV.

[16] Bradski, G. (2000). The OpenCV library. Dr. Dobb's Journal of Software Tools.

[17] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

[18] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020).

[19] Waskom, M. L., (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021, https://doi.org/10.21105/joss.03021.

Source code: https://github.com/NicKylis/SveltNet