Advanced Deep Learning Techniques: Classification and Feature Extraction

# Introduction

The use of advanced algorithms in contemporary data science has proven critical in revealing deep patterns hidden within datasets (Sarker 2021). More specifically, deep learning (DL) has garnered massive attention in the world of computing, due to its widespread application in healthcare, computer vision, sentiment analysis, and more (Sarker 2021). In this report, we aim to uncover how sophisticated DL approaches—convolutional neural networks (CNNs) and autoencoders (AEs) can be used to tackle image classification tasks.

## Our primary objectives include the implementation of a CNN for flower class prediction and the utilization of AEs for unsupervised feature extraction in digit recognition.

In the first part of our analysis, we train a CNN to predict flower type (i.e., daisy, rose, tulips, etc.) within a given dataset. Additionally, our exploration includes experimenting with different architectures and altering the convolutional layers to improve prediction accuracy. For our digit recognition portion, we focus on training AEs sequentially to extract relevant features from images of handwritten digits. Then these learned representations are utilized in constructing a deep neural network (DNN) for accurate digit classification, followed by fine-tuning to optimize predictive performance.

This analysis offers a comprehensive examination of several tasks, covering methodologies, experimental setup, and results. Our paper emphasizes the reasoning behind certain practices and offers a complete analysis of the results obtained. Considering this, we hope to contribute to the ongoing discourse regarding data science approaches by demonstrating the power of complex DL algorithms in classification tasks and feature extraction.

# Materials & Methods

## CNN Multi-class Classification

### 1.1 Dataset Collection & Preprocessing

The flower dataset we used contained 3670 images of five different flower classes: daisies, dandelions, roses, sunflowers, and tulips. We evaluated our model using a validation split of 20% (n=731) with the remaining data (n=2939) used for training. Before training our model we first scaled image pixels from a range of 0-255 to a range of 0-1. Data augmentation techniques—scaling, sheering, zooming, and horizontal flips—were applied to introduce variability and enhance model robustness (Saboo et al., 2023). Together these preprocessing steps will help speed up computation and improve model performance (Saboo et al., 2023).

### 1.2 Model Construction & Architecture

To build our first CNN we first performed a grid search of three different dropout rates—0.1, 0.3, and 0.5. Then using the Keras framework, we built our first baseline CNN model with three convolutional layers, each fitted with 32, 64, and 64 filters, respectively. The kernel size was the same for each layer with dimensions of 3x3 and padding was applied to maintain spatial resolution. In addition, each convolutional layer was followed by ReLU activation and an average/max pooling layer. Finally, we applied our dropout using the best dropout rate determined by our initial grid search and finished our model with a predictive SoftMax layer.

To understand the relationship between convolutional layers and model performance we constructed three different CNNs containing either 2,3, or 4 convolutional layers. The model's primary components are as follows:

1. **Convolutional Layers:** Convolutional layers were organized such that the first layer for each CNN had 32 filters and subsequent filters were doubled to mimic increasing complexity. Each layer was also fitted with a 3x3 kernel irrespective of the number of filters. Additionally, each convolutional layer was followed by a ReLU activation and average/max pooling operations.
2. **Dense Layers:** For each CNN we flattened the output of the last convolutional layer before feeding into our dense layer comprising with ReLU activation. This dense layer was followed by a dropout layer using our grid search dropout rates and then a SoftMax layer for classification.
3. **Regularization:** We used dropout regularization.

### 1.3 Training and Optimization

We compiled our model using the Adam optimizer provided by Keras. The Adam optimizer provided by Keras is a stochastic gradient descent approach based on the adaptive estimate of first- and second-order moments ("Keras Documentation," 2023). This optimizer is highly effective because it is computationally efficient, requires little memory, and is insensitive to diagonal rescaling of gradients (Kingma et al., 2014). Since each flower class is a separate folder, we measured our training progress using the categorical cross-entropy. The CNN was trained with a batch size of 32 images and an image size of 64x64 pixels. The model was trained over 20 epochs, and progress was monitored by recording the training and validation loss across epochs.

## Autoencoder Feature Extraction

### 2.1 Data Collection & Preprocessing

In the next portion, we used a subset of the MNIST dataset which contains images of handwritten digits. We chose 5000 images each for the train and test sets. Similar to our CNN task we normalized our image pixels to a range of 0-1.

### 2.2 Model Construction & Architecture

#### Autoencoder 1

We composed the first autoencoder with six dense layers, three encoding layers and three corresponding decoding layers. The encoding layers were organized with 784,64, and 32 nodes while the decoding layers had the ascending order of 64,128 and 784 nodes. All layers used a ReLU activation except for the last decoding layer which utilized the sigmoid activation. The final model was compiled with an Adam optimizer and training progress was measured by computing binary cross entropy. The aim of our first autoencoder was to learn a compressed representation of the handwritten digits.

#### Autoencoder 2

The second autoencoder used the output from our first autoencoder as input. The second autoencoder consisted of the same layer organization as the decoder in the first autoencoder (64, 128, and 784 nodes). Autoencoder 2 also used the Adam optimizer, and the training progress was measured by calculating the mean squared error. This second autoencoder was built to further refine the extracted features from the first autoencoder.

#### SoftMax Layer

The SoftMax layer is trained similarly to our first two autoencoders, however, we measure progress using the sparse categorical cross entropy function rather than mean squared error. The SoftMax layer takes the flattened output from the second encoder to predict digit classes.

#### Deep Neural Network

Lastly, we constructed a DNN by using our first and second autoencoders as hidden layers, leading to a final SoftMax layer.

### 2.3 Training, Optimization, and evaluation

Autoencoders 1 & 2 were trained for 500 epochs with a batch size of 32. While The SoftMax layer was trained similarly with a validation split of 0.2 (n=1000). After building and compiling our DNN, we performed an initial evaluation using the test set (n=5000). This evaluation provided a baseline metric for how well our model was performing before hyperparameter tuning. Subsequently, we retrained the model over 25 epochs and a batch size of 128 images. The recorded metrics included accuracy and loss values for both the training and validation datasets.

# Results

## CNN Classification

Table 1 Grid search results measuring three different dropout rates.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Configuration | Filters | Epochs | Dropout rate | Validation Accuracy | Validation Loss |
| 3 Convolutional layers | 32,64,64 | 50 | 0.1 | 0.6875 | 0.9043 |
| 3 Convolutional layers | 32,64,64 | 50 | 0.3 | 0.6449 | 0.9544 |
| 3 Convolutional layers | 32,64,64 | 50 | 0.5 | 0.7031 | 0.8282 |

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

Figure 1 Baseline model performance based on best grid search drop out rate.

Table 2 Validation results for three different CNN configurations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number of Convolutional Layers | Dropout | Filters | Validation Accuracy | Validation Loss |
| 2 | 0.5 | 32,64 | 0.6918 | 0.8485 |
| 3 - Baseline | 0.5 | 32,64,64 | 0.7031 | 0.8282 |
| 4 | 0.5 | 32,64,64,128 | 0. 6889 | 0.8203 |

A graph of different colored lines

Description automatically generated with medium confidence

Figure 2 Model performance for two convolutional layer configuration.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

Figure 3 Model performance for four convolutional layer configuration

A group of white letters on a black background

Description automatically generated

Figure 4 Original images located on the top and reconstructed images on the bottom row.

Table 3 Model performance results for each layer of our deep neural network

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Configuration | Loss | Epochs | Validation Loss | Validation Accuracy |
| Autoencoder 1 | Binary Cross Entropy | 500 | 0.0730 | - |
| Autoencoder 2 | Mean Squared Error | 500 | 0.0040 | - |
| SoftMax Layer | Sparse Categorical Cross Entropy | 150 | 0.0030 | 0.9010 |
| Deep Neural Network (Before Training) | Sparse Categorical Cross Entropy | - | 2.3781 | 0.047 |
| Deep Neural Network (After training) | Sparse Categorical Cross Entropy | 30 | 0.3620 | 0.9204 |

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

Figure 4 Model performance for our DNN

# Discussion

## CNN Classification

To answer our first objective, we were tasked with creating a CNN with a SoftMax layer to predict flower class. To aid in constructing my CNN I performed a grid search of different dropout rates (i.e. 0.1, 0.3, 0.5). The model achieved the best validation accuracy of 70.3% with a dropout rate of 0.5. These results suggest that a higher dropout rate might be more beneficial to increasing accuracy and reducing overfitting. Further, these findings align with current literature that has found the ideal dropout rate to be usually around 0.5 (Srivastava et al., 2014; Saboo et al., 2023). Controlling your dropout rate is especially important in DL, as large neural networks tend to have many hyperparameters and a high variance. Therefore, dropout is a critical regularization method to prevent the model from overly relying on one set of neurons or a single pattern (Saboo et al., 2023). However, it is also important to note that while dropout plays a vital role in CNN construction, higher dropout rates might introduce more sparsity, thus potentially leading to reduced model performance.

Manipulating the dropout rate does not reveal the entire story about model performance, so in the second part of this task, we experimented with various CNN architectures. While the main components of the CNN remained the same, we changed the number of layers for three CNNs. Our baseline CNN architecture built with three convolutional layers had the highest accuracy of 70.3% compared to 69.2% accuracy for two convolutional layers and 68.9% for four. These findings imply that simply adding more convolutional layers does not inherently improve model performance (Hu et al., 2019). This trend aligns with the notion that overly complex models fail to learn subtle patterns from a small sample (Hu et al., 2019). While four convolutional layers increased model complexity which may mean more learning opportunities, it did not necessarily translate into high accuracy. For this reason, three convolutional layers appear to strike a balance between model complexity and generalization, leading to a better classification model (Hu et al., 2019).

## AE Classification

AEs have become especially relevant with the innovation of generative AI as they are robust tools for learning representations from unlabelled data (Singh & Ogunfunmi 2022). In this study, we first trained an autoencoder on a set of handwritten digits to automatically extract features that are good predictors of the digit class. Aside from minor inconsistencies in blurriness seen in our reconstructed images, it is clear our AE effectively compressed the input images while preserving essential features. These observations are consistent with previous research illustrating the effective nature of AEs to capture salient information from high-dimensional data, a key element in many areas such as image processing, text, image, and audio generation (Singh & Ogunfunmi 2022).

Interestingly, when training the first AE I found that batch size had the biggest influence on reconstruction quality. More specifically, when we used a higher batch size (e.g. 64 or 128) the reconstructed images were extremely blurry and not entirely reflective of the original image. However, by reducing the batch size to 32, we see the well-reconstructed images shown in *Figure 4*. This discovery implies that lower batch sizes may have permitted a more refined learning process, resulting in improved feature extraction and, as a result, increased reconstruction accuracy. Moreover, these results are supported by Kerley et al. 2023 research as they also found smaller batch sizes tended to yield better accuracy overall. The authors posit that smaller batch sizes could allow the model to focus on finer details, avoid poor local minima, and reduce the impact of noise (Kerley et al., 2023). The autoencoder's sensitivity to batch size changes emphasizes the relevance of this hyperparameter in improving the model's performance. Moreover, it highlights the necessity for careful study and consideration of batch size settings in comparable unsupervised learning tasks (Kerley et al., 2023).

To further refine our model, we employed the sequential application of multiple AEs to reflect current advancements in hierarchical feature learning. The first AE compressed the input data, while the second AE further improved this feature extraction allowing the network to learn more intricate representations of the input data (Ferri et al., 2021). Finally, we transitioned from encoded features to multi-class classification by implementing a SoftMax layer. The encoded features displayed substantial discriminative power, achieving a validation accuracy of 90%. This observation not only reaffirms the potential of unsupervised feature learning but also highlights the capacity for autoencoders to generate meaningful and transferable representations suited for classification tasks.

To complete our autoencoder experimentation we used our previously built autoencoder and SoftMax layer to create, train, and test a deep neural network (DNN). This cascaded design of our stacked autoencoders culminating in a predictive SoftMax layer is like the progressive learning paradigms used in advanced DL approaches. While our initial untrained DNN yielded a measly accuracy of 4.7% retraining led to a significant performance boost of 92.1% accuracy. Evidently, the iterative refinement of representations increases the adaptability and learning capacity of DNNs when given feature rich.

# Conclusion

The results from this study underscore the importance of fine-tuning model architectures and hyperparameters achieve optimal performance in classification tasks. Our CNN observations revealed that while deeper networks are theoretically better at feature discrimination, their efficacy is limited by data quantity. This implies that researchers and engineers working with these models should carefully consider how to best enhance generalization without sacrificing predictive performance. As demonstrated in our study, regularization is great technique and experimenting with different drop out rates can improve model variance. In regard to, autoencoders its integration into DNNs for image classification tasks demonstrated in this paper are consistent with existing work, which emphasizes the potential for hierarchical feature learning and transferability of learnt representations for improved model performance.

# Limitations & Future Directions

This study acknowledges limitations including the use of only a singular dataset for training and evaluating both neural networks. Moreover, the evaluation of a limited number of hyperparameters and architectures. Future research could expand on these findings by exploring additional model architectures and evaluating performance on diverse datasets to enhance the credibility of our findings. Lastly, it would be interesting to evaluate applications of autoencoders in domains beyond image classification.

References

Ferri, R., Babiloni, C., Karami, V., Triggiani, A. I., Carducci, F., Noce, G., Lizio, R., Pascarelli, M. T., Soricelli, A., Amenta, F., Bozzao, A., Romano, A., Giubilei, F., Del Percio, C., Stocchi, F., Frisoni, G. B., Nobili, F., Patanè, L., & Arena, P. (2021). Stacked autoencoders as new models for an accurate Alzheimer’s disease classification support using resting-state EEG and MRI measurements. *Clinical Neurophysiology*, *132*(1), 232–245. https://doi.org/10.1016/j.clinph.2020.09.015

Hu, W., Zhang, Y., & Li, L. (2019). Study of the Application of Deep Convolutional Neural Networks (CNNs) in Processing Sensor Data and Biomedical Images. *Sensors*, *19*(16), 3584. https://doi.org/10.3390/s19163584

Kerley, C. I., Cai, L. Y., Tang, Y., Beason-Held, L. L., Resnick, S. M., Cutting, L. E., & Landman, B. A. (2023). Batch size go big or go home: counterintuitive improvement in medical autoencoders with smaller batch size. In I. Išgum & O. Colliot (Eds.), *Medical Imaging 2023: Image Processing* (p. 17). SPIE. <https://doi.org/10.1117/12.2653643>

Kingma, D. P., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization*.

Saboo, Y. S., Kapse, S., & Prasanna, P. (2023). Convolutional Neural Networks (CNNs) for Pneumonia Classification on Pediatric Chest Radiographs. *Cureus*. https://doi.org/10.7759/cureus.44130

Sarker, I. H. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Computer Science*, *2*(6), 420. https://doi.org/10.1007/s42979-021-00815-1

Singh, A., & Ogunfunmi, T. (2021). An Overview of Variational Autoencoders for Source Separation, Finance, and Bio-Signal Applications. *Entropy*, *24*(1), 55. https://doi.org/10.3390/e24010055

Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.

TensorFlow. (n.d.). Adam (tf.keras.optimizers.Adam). TensorFlow. Retrieved Dec 8, 2023, from https://keras.io/api/optimizers/adam/