Deep Learning Character Classification

Abstract

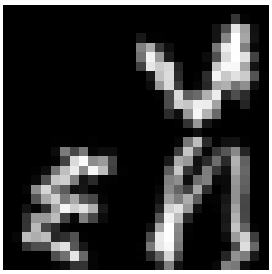
This study presents an investigation into the efficacy of different ResNet architectures for a complex character recognition task using a dataset combining Japanese numerals and EMNIST characters. The dataset, consisting of grayscale bitmap images, was preprocessed through resizing for compatibility with the ResNet model. We experimented with four ResNet variants (ResNet50, Modified ResNet50, ResNet152, Modified ResNet152) and assessed their performance using criteria like validation and test accuracy, and computational efficiency. Our fin dings reveal that the standard ResNet50 model outperformed others, suggesting that increased model complexity does not always yield better results for specific tasks and datasets.

1 Introduction

For image classification, Convolutional Neural Networks (CNNs) have emerged as a powerful tool, with ResNet (Residual Network) being a prominent example [9]. This project explores the use of various ResNet architectures for classifying a unique dataset combining Japanese numerals and EMNIST characters. The dataset, comprising low-resolution grayscale images, poses a challenge for character recognition tasks. Preprocessing involved resizing these images to suit the ResNet model. We compared four variants of ResNet (ResNet50, Modified ResNet50, ResNet152, Modified ResNet152) to determine the most efficient and accurate model for this specific task. The study's focus is on evaluating the performance of these models in terms of accuracy and computational efficiency, highlighting the impact of model complexity on task-specific performance.

2 Dataset

The dataset of this project comprises a collection of images, each featuring a Japanese digit alongside characters from the EMNIST dataset[1] (which includes handwritten digits and letters of the English alphabet). This combination of Japanese numerical symbols and diverse EMNIST characters creates a dataset suitable for complex character recognition tasks. Each image in the training data has a corresponding label that identifies the Japanese numeral, with labels ranging from 0 to 9 for a 10-class classification system. Each sample is a grayscale bitmap image in a pixel-based format (28 * 28), where the values of the pixels (features) are normalized (between 0 and 1). The relatively low resolution of the images suggests that the dataset may include a degree of difficulty to challenge the robustness of our recognition algorithms.



The preprocessing procedure of the data had two stages:

1. Squeeze: Since the original images have a dimension of (1, 28, 28), due to their grayscale nature, we remove the single dimension to perform the resizing in the next step.
2. Resizing: We performed the resizing of the images from 28x28 to 224x224 pixels to adapt them for use with the ResNet model that we chose for the classification task. By enlarging the images, we ensured compatibility with the model's architecture, enabling us to leverage ResNet's powerful feature extraction capabilities. To be able to minimize the potential quality loss or artifacts introduced due to resizing, we have tested different methods of Interpolation. Interpolation in the context of image processing is a mathematical method used to estimate new pixel values for locations in a transformed image (in this case, enlarged images). Here are some of the approaches that we have tested for our dataset:
   (a) Nearest-neighbor Interpolation: This is the simplest form of interpolation. It assigns to a new pixel the value of the closest pixel in the original image. While fast, it can result in a blocky or jagged appearance, especially when enlarging images [8].
   (b) Bilinear Interpolation: This method takes a weighted average of the four nearest pixels to estimate a new pixel value. It provides smoother results than nearest-neighbor interpolation. It's a good balance between quality and performance [2].
   (c) Bicubic Interpolation: Bicubic interpolation considers the 16 nearest pixels (a 4x4 area) to estimate a new pixel value. it's computationally more intensive than the previous approaches [10].
   (d) Lanczos Interpolation: This is a high-quality resampling method for geometric transformations of images. It uses a convolution of the input image with a Lanczos kernel [7].
   (e) Area-based (or Pixel Area Relation) Interpolation: In this method, pixel values are computed based on the area of overlap of the pixel with the original image pixels. This approach is especially effective for reducing the size of images (downsampling), as it helps to preserve the overall appearance and reduces aliasing effects [3].

   Eventually we have used Bilinear Interpolation as it maintained smooth gradients and avoided artifacts. The effect of each image can be seen in the figure 2.

3 Proposed Approach

Convolutional Neural Networks (CNNs) are typically considered the best choice for image classification tasks due to several key characteristics [5]:

- Local Connectivity
- Shared Weights and Translation Invariance
- Hierarchical Feature Learning
- Reduction of Parameters through Pooling
- Robustness to Image Transformations

The structural design of CNNs aligns well with the nature of image data, enabling them to learn patterns effectively, handle variations in input, and perform well even with relatively less training data compared to fully connected networks. This makes them the go-to choice for image classification tasks. ResNet, short for Residual Network, is a type of convolutional neural network (CNN) that was

introduced by researchers from Microsoft Research in their paper "Deep Residual Learning for Image Recognition"[4]. ResNet is renowned for enabling the training of extremely deep neural networks and is one of the most influential innovations for tasks like image recognition. In a typical deep neural network, each layer learns a representation of the data based on the output of the previous layer. However, as the network gets deeper, this process can lead to the vanishing gradient problem, where the gradients used in training the network become so small that they effectively prevent the network from learning effectively. Residual blocks which are a key component of the ResNet architecture, address this problem by introducing what's known as a "skip connection" or "shortcut connection." This connection allows the input of the residual block to be added directly to the output of a layer a few steps further along. The main advantages of this design are:

- Easing the Training of Deep Networks: The skip connections allow the gradient to flow directly through the network without being dampened by deep layers, making it easier to train very deep networks and learn from a vast range of features.
- Improved Learning: By learning the residual (difference) rather than the full output, each layer needs only to learn the part of the representation that has not been captured by previous layers. This often leads to more efficient learning.
- Avoiding the performance degradation Problem: In very deep networks without residual connections, performance tends to plateau or even degrade as depth increases. Residual blocks help to avoid this problem by making it easier for deeper networks to at least match the performance of shallower ones.

3.1 Model Selection

We employed four variants of ResNet:

1. ResNet50 with No Modification: The architecture of ResNet50 consists of 50 layers. ResNet50 starts with a convolutional layer followed by a max-pooling layer. The output then passes through a series of residual blocks, each consisting of convolutional layers, batch normalization, and ReLU activation functions. The unique aspect of these blocks is the addition of the input of the block to the output of the block before passing through another ReLU function. The final layer is a fully connected layer, mapping the output of the last residual block to the output classes. For this model, we only changed the structure of input and output to accommodate our model with our data. The original ResNet50 model is configured for RGB images, which means it accepts inputs with three channels (red, green, and blue). We altered the first convolutional layer of the ResNet50 model so that it would accept one-channel gray-scale input. Moreover, the standard ResNet50 model outputs a 1000-class prediction, aligning with the ImageNet dataset used in its original formulation. Therefore, we replaced the final fully connected layer of the ResNet50 model with a new fully connected layer that outputs ten classes.
2. In another approach to optimizing the ResNet50 model for our 10-class classification task, we introduced another adaptation by adding two additional fully connected layers toward the end of the network. This modification was used to mitigate the substantial reduction in the output class size, from the original 1000 classes to 10. The first newly added fully connected layer effectively reduces the parameter count from 100352 (when flattened) to 1024. Following this, the second layer further decreases the parameters from 1024 to 10. This two-step reduction process in the network's architecture ensures a more gradual and refined transition from the high-dimensional feature representations to the final class predictions. Gradually reducing the dimensionality allows the network to maintain and refine the most relevant features for classification, leading to better learning and generalization.
3. Resnet152 with no Modification: Another model employed in our project was ResNet152. ResNet152 is an extended version of the ResNet model, containing 152 layers, which makes it one of the deepest architectures in the ResNet family. Like its counterparts, ResNet152 is built on the principle of deep residual learning, but it offers a more profound and complex network structure due to its increased number of layers. The motivation behind using ResNet152 in this context is its enhanced capacity for feature extraction and learning. With more layers, ResNet152 can learn a broader range of features at various levels of abstraction, which is particularly beneficial for complex image classification tasks. The depth of ResNet152 allows it to capture intricate patterns in data that might be missed by

shallower networks such as ResNet50. For this training, we only changed the structure of input and output to accomodate our model with our data as mentioned before.

4. Resnet152 with Additional Layers: We added two additional fully connected layers similar to what we did for Resnet50 with the same motivation as mentioned.

3.2 Loss Criterion

We used Cross-Entropy as the loss function which is effective in measuring the difference between two probability distributions - the actual labels and the predictions. This makes it suitable for training models where the output is a probability distribution such as multi-class classification.

3.3 Hyperparameters Selection

1. **Learning Rates**: We used Adam optimizer[6], short for "Adaptive Moment Estimation", which computes adaptive learning rates for each parameter during training. Adam is often more computationally efficient than some other optimization methods and can be beneficial for models with a large number of parameters and datasets with noisy or sparse gradients.
2. **Batch size**: We've chosen a batch size of 64 for our DataLoader. Larger batch sizes can lead to faster training (due to parallelism and optimization of hardware usage), but sometimes at the cost of generalization. A batch size of 64 is often large enough to benefit from the computational efficiencies of vectorized operations, while still being small enough to maintain a degree of noise in the training process, which can aid in generalization. Our choice of batch size was also constrained by hardware limitations. A batch size of 64 was the maximum manageable size for Google Colab environment while training the models.
3. **Kernel size**: We have used the 7*7 kernel to balance the capture of detailed and complex spatial information with computational efficiency.
4. **Number of Epochs**: To determine the proper number of epochs, we have trained the models with different numbers of epochs (progressively increasing), until the loss reached less than a predefined threshold (0.01) to ensure sufficient learning while avoiding unnecessary computations. This approach also helps in preventing both underfitting (by ensuring sufficient training) and overfitting (by stopping training once the desired performance is reached). For the Resnet 50-based models the proper number was 20 and for Resnet152-based ones, it was 30 epochs. The threshold for the loss also was chosen by calculating the accuracy of the model with different loss values on the validation set, and found that this threshold (0.01) provides a good accuracy (look at figure 3).

```
750/750 [==============================] - 4934s 7s/step - loss: 6.0542 - accuracy: 0.2500
Epoch 2/10
750/750 [==============================] - 4921s 7s/step - loss: 0.8243 - accuracy: 0.7322
Epoch 3/10
750/750 [==============================] - 4915s 7s/step - loss: 0.3900 - accuracy: 0.8755
Epoch 4/10
750/750 [==============================] - 4916s 7s/step - loss: 0.2616 - accuracy: 0.9162
Epoch 5/10
750/750 [==============================] - 4911s 7s/step - loss: 0.1963 - accuracy: 0.9368
Epoch 6/10
750/750 [==============================] - 4907s 7s/step - loss: 0.1650 - accuracy: 0.9462
Epoch 7/10
750/750 [==============================] - 4911s 7s/step - loss: 0.1442 - accuracy: 0.9527
Epoch 8/10
750/750 [==============================] - 4906s 7s/step - loss: 0.1280 - accuracy: 0.9580
Epoch 9/10
750/750 [==============================] - 4907s 7s/step - loss: 0.1175 - accuracy: 0.9613
Epoch 10/10
750/750 [==============================] - 4886s 7s/step - loss: 0.1016 - accuracy: 0.9671
```

4 Results

ResNet50 outperforms the other models on both the validation and test sets, indicating its effectiveness and generalization capability for our dataset and task. The high accuracy on both sets suggests that the model was well-tuned and did not suffer from overfitting. ResNet152, despite its

additional depth, does not surpass ResNet50 in performance and has a similar result (within the error range). This indicates that the added complexity of ResNet152 is not necessary for the dataset, potentially leading to inefficiencies in learning, or that the ResNet50 is complex enough to have a good performance for our dataset. Both modified versions of ResNet50 and ResNet152 show a drop in performance compared to the original models. This decline could be due to the added layers introducing unnecessary complexity or not being optimized for our specific task. Moreover, ResNet50's of 92.76% on the test set demonstrates its suitability for the task.

Model	**Validation Accuracy (%)**	**Test Accuracy (%)**	**Runtime**
ResNet50	95.7	92.76	$\sim$2 hours
ResNet152	92.5	90.5	$\sim$5 hours
Modified ResNet50	91.0	89.16	$\sim$2 hours
Modified ResNet152	90.1	89.0	$\sim$5 hours

We used the same batch size the models. However, the runtime for ResNet152-based models was more than twice as long as that for ResNet50 ones. ResNet152 is significantly deeper than ResNet50. Each additional layer adds computational complexity and requires more operations. Also, ResNet152 has a larger number of trainable parameters compared to ResNet50. Besides, deeper networks can sometimes have slower convergence rates and may require more careful tuning of hyperparameters. Eventually, the number of epochs for the ResNet152 was larger than ResNet50 to achieve the same loss. The efficiency of ResNet50, in terms of both computational resources and time, makes it a more attractive choice when considering the balance between performance and resource utilization. The consistency in performance trends between validation and test sets suggests effective model selection and tuning. However, the results also indicate that increased complexity in deep learning models does not always translate to better performance, especially when considering computational efficiency and the specific nature of the task and dataset. Following this, we placed on the 6th position in the kaggle leader-board.

5 Discussion and Conclusion

The research concludes that the standard ResNet50 model, without any modifications, demonstrates superior performance in terms of accuracy and computational efficiency for the specific task of recognizing a combination of Japanese numerals and EMNIST characters. This finding challenges the assumption that increased complexity in deep learning models invariably leads to better results. The study's results suggest that simpler models like ResNet50 can be more effective and efficient than their more complex counterparts for certain types of datasets and tasks. Future work could involve fine-tuning hyperparameters, exploring different layer modifications, and optimizing GPU usage to enhance model performance further. This study contributes to the broader understanding of the relationship between model complexity and task-specific performance in the field of image classification. For future improvements, we suggest the following:

1. Fine-tuning Hyperparameters (such as batch size, or optimizer settings)
2. Layer or Skip Connection Modification
3. Optimizing GPU usage and parallel processing to make training more efficient
4. Increasing the number of epochs (carefully to prevent overfitting)
5. Extracting custom feature (tailoring the early layers specifically for the types of features most relevant to the data

Additionally, for the modifed versions of Resnet50 and 152, modifying the additional layers to ensure the modifications don't excessively increase the model's complexity without a corresponding benefit in performance and adding specialized layers that target specific aspects of the data.

References

[1] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre van Schaik. Emnist: an extension of mnist to handwritten letters, 2017.

[2] K.T. Gribbon and D.G. Bailey. A novel approach to real-time bilinear interpolation. In *Proceedings. DELTA 2004. Second IEEE International Workshop on Electronic Design, Test and Applications*, pages 126–131, 2004.

[3] Guo, Chih-Yuan Hsu, Guo-Ching Shih, and Chia-Wei Chen. Fast pixel-size-based large-scale enlargement and reduction of image: adaptive combination of bilinear interpolation and discrete cosine transform. *Journal of Electronic Imaging*, 20(3):033005, 2011.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Md Anwar Hossain and Md Shahriar Alam Sajib. Classification of image using convolutional neural network (cnn). *Global Journal of Computer Science and Technology*, 19(2), 2019.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] B. N. Madhukar and R. Narendra. Lanczos resampling for the digital processing of remotely sensed images. In Veena S. Chakravarthi, Yasha Jyothi M. Shirur, and Rekha Prasad, editors, *Proceedings of International Conference on VLSI, Communication, Advanced Devices, Signals & Systems and Networking (VCASAN-2013)*, pages 403–411, India, 2013. Springer India.

[8] Vaishali Patel and Kinjal Mistree. A review on different image interpolation techniques for image enhancement. *International Journal of Emerging Technology and Advanced Engineering*, 3(12):129–133, 2013.

[9] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019.

[10] Cheng Yun, Wang Yan, and Liu Qi. Research on digital image scaling based on bicubic filter algorithm. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pages 225–229, 2018.