

Project Report

Image Classification with CIFAR-100

Robert Veloya

Robert Veloya

December 3, 2023

I. Introduction

Machine learning is a rapidly growing field, where image classification stands out as an imperative area of research and application. The concept of image classification presents its own significant opportunities in the advancement of deep learning, as well as its challenges especially with highly complex datasets such as CIFAR-100 which is known for its diversity and depth.

CIFAR-100 is a highly recognized dataset, created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton, and is a part of the Canadian Institute For Advanced Research (CIFAR). This dataset is not only proof of the ever-growing field of machine learning, but it also serves as a tool that challenges and pushes the boundaries of image classification algorithms. It is composed of 60,000 32x32 color images that are distributed across 100 classes where it receives its name. The dataset is composed of a wide array of images that range from commonplace images to more abstract concepts.

This project dives into the exploration of advanced deep learning algorithms as it uses its own Convolutional Neural Networks Model, a deep learning algorithm that is predominantly used for processing data with a grid-like topology. Since the nature of this project revolves around the CIFAR-100 dataset, the aforementioned deep learning algorithm is utilized. This study aims to not only develop models that can accurately categorize images into the 100 classes of the dataset, but it also aims to contribute to the broader understanding of deep learning and its optimization and application for such tasks. This study, not only as a technical endeavor, but also an academic contribution, embodies the pursuit of knowledge in the field of machine learning that focuses on image classification algorithms.

II. Literature Overview

The foundation of this study revolves around the seminal external works that shared the field of image classification. *ImageNet Classification with Deep Convolutional Neural Networks*, a paper made by Krizhevsky et. al.(2012), covers the development of the AlexNet model, a CNN that was able to efficiently classify high-resolution images from the ImageNet database, and set the standard in accuracy which outperformed traditional image classification methods. Not only did the study demonstrate the potential of CNN's, but it also laid the foundation for future research in neural network design.

He et al. (2016) introduced a novel that pushed the boundaries even further. In their paper *Deep Residual Learning for Image Recognition*, introduced ResNet, a novel architecture that enabled the training of networks to go deeper, by leveraging skip connections to jump over some layers that were addressed by the vanishing gradient problem. This proved to be a stepping stone towards the right direction that enhanced the ability of neural networks to learn from a vast amount of data without compromising the depth of complexity of the dataset.

Densely Connected Convolutional Networks, a CVPR paper contributed by Huang et al. (2017), contributes to the development of DenseNet, an architecture that presents the idea of having each layer connected to every other layer in a feed-forward fashion. The inventive approach not only optimized information flow in the network, but it also reduced the number of parameters leading to the reduction of difficulty to train the network. DenseNet's architecture displayed the improvements in image classification tasks by emphasizing the efficiency and reduction of parameters while leaving performance levels untouched.

III. Methodology

Datasets and Preprocessing

The project utilizes the CIFAR-100 dataset, which consists of 60,000 32x32 color images across 100 classes. Each class in the dataset consists of 600 images, where 500 of them are used for training and

the remaining 100 for testing. As stated in the code provided along with the study, the images are normalized by dividing by 255. This normalization step is crucial for deep learning models as it aids in the convergence of the training process as it scales the pixel values to a range of 0 to 1.

Data Augmentation

Data Augmentation techniques were applied to the CNN models in order to enhance their ability to generalize and reduce overfitting. These techniques artificially expand the dataset by creating modified versions of the training images, providing the models with more varied data to learn from. From the code provided along with the study, ImageDataGenerator was used for data augmentation, utilizing a rotation range of 20 degrees, horizontal flipping, a zoom range of 0.2, and width and height shift of 0.2

Model Architectures

A. Custom CNN Model

A custom CNN model was developed for the objective of this project. The model architecture is defined using a Sequential model from Keras. The CNN model includes convolutional layers with different filter sizes and kernel sizes, followed by Batch Normalization, MaxPooling, as well as Dropout Layers. Specifically, Figure 3.1.a shows the different layers of the custom CNN model, showcasing the 11 different layers used.

```
def __init__(self, input_shape):
    self.input_shape = input_shape

def build(self, hp):
    model = Sequential()
    model.add(Conv2D(filters=hp.Int('conv_1_filter', min_value=32, max_value=128, step=32),
                    kernel_size=hp.Choice('conv_1_kernel', values=[3, 5]),
                    activation='relu',
                    input_shape=self.input_shape))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(2, 2))
    model.add(Dropout(rate=hp.Float('dropout_1', min_value=0.0, max_value=0.5, step=0.1)))

    model.add(Conv2D(filters=hp.Int('conv_2_filter', min_value=32, max_value=128, step=32),
                    kernel_size=hp.Choice('conv_2_kernel', values=[3, 5]),
                    activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(2, 2))
    model.add(Dropout(rate=hp.Float('dropout_2', min_value=0.0, max_value=0.5, step=0.1)))

    model.add(Flatten())
    model.add(Dense(units=hp.Int('dense_units', min_value=32, max_value=512, step=32),
                    activation='relu'))
    model.add(Dense(100, activation='softmax'))

    model.compile(optimizer=SGD(learning_rate=0.01, momentum=0.9),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

Figure 3.1.a: Different Layers Used in the Custom CNN Model | Source: Author

It can also be observed from Figure 3.1.a, the hyperparameter tuning technique used from the HyperModel class and RandomSearch from Keras Tuner. The model undergoes 5 trials of hyperparameter tuning in order to optimize the model's performance.

B. ResNet-50

Additionally, the study also incorporates the ResNet50 architecture. The pre-trained model is utilized with weights from ImageNet, and a GlobalAveragePooling2D and a Dense Layer are added for classification of the dataset. Figure 3.1.b below shows the 2 layers that were implemented in the ResNet50 model.

```
# ResNet50 model setup and training
resnet_base = ResNet50(weights='imagenet', include_top=False, input_shape=(32, 32, 3))
resnet_model = Sequential()
resnet_model.add(resnet_base)
resnet_model.add(GlobalAveragePooling2D())
resnet_model.add(Dense(100, activation='softmax'))

resnet_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
resnet_model.fit(train_images, train_labels, epochs=10, validation_data=(test_images, test_labels))

resnet_predictions = resnet_model.predict(test_images)
resnet_predictions_classes = np.argmax(resnet_predictions, axis=1)

resnet_report = classification_report(test_labels, resnet_predictions_classes, target_names=['Class'+str(i) for i in range(100)])
print('ResNet50 Model Classification Report:\n', resnet_report)
```

Figure 3.1.b: ResNet50 setup | Source: Author

Training and Evaluation

Both models are trained and validated with the CIFAR-100 Dataset. The custom CNN model is trained with data augmentation, utilizing a batch size of 64 over 20 epochs. The pretrained model, ResNet50, on the other hand, is trained with 10 epochs due to the hardware limitations encountered during the facilitation of the study; however, the ResNet50 model utilizes sparse categorical cross entropy loss and tracking accuracy as a metric.

IV. Implementation

The implementation of this project involved training and testing two different models, a Custom CNN model, and a pretrained ResNet50 model to classify images from the CIFAR-100 dataset.

Custom CNN Model

The custom CNN model was built using TensorFlow and Keras. The architecture of the model was defined by the Sequential model, which included multiple convolutional layers. The implementation involved the specification of the number of filters, kernel sizes, and the incorporation of Batch Normalization, Max Pooling, and Dropout layers in order to enhance performance and address overfitting from the project's initial implementations. An Stochastic Gradient Descent optimizer was used for compiling and trained on the CIFAR-100 dataset for over 20 epochs.

ResNet50 Model

The ResNet50 model, with power of transfer learning, was employed along with the custom CNN model. ResNet50 from Keras applications was utilized with pretrained ImageNet weights. The model utilized 2 layers: GlobalAveragePooling2D and a Dense Layer were added for CIFAR-100 classification. An Adam optimizer was used for compiling the model and it was trained for over 10-epochs due to hardware limitations.

Both models were assessed with the use of a Classification Report from the scikit-learn library, printing out the precision, F1-score, recall, and accuracy of both models for each of the 100 classes of the dataset.

V. Experimental Setup

The experimental setup for this project was centered around the evaluation of the two models. The design involved training these models on the dataset, followed by a validation phase using the testing set embedded into the dataset. Specific focus was placed on adjustment and optimization of the models to effectively handle the diverse range of classes within the CIFAR-100 dataset.

CNN Model:

The architecture of the CNN model consisted of a total of 11 layers:

- Two Convolutional 2d Layers: used to create a convolution kernel that is convolved with the layer input to produce a tensor of outputs

- BatchNormalization Layers: normalizes the activation of the previous layer at each batch, applying a transformation that maintains the mean activation to 0 and the activation standard deviation close to 1.
- MaxPooling 2D layers: reduces the spatial dimensions of the input volume. Helping in the reduction of number of parameters and computation in the network in hopes of addressing overfitting.
- Dropout Layers: randomly set a fraction rate of input units at each update during the training time, also in hopes of addressing overfitting.
- Flatten Layer: flattens the input without affecting the batch size. Used to flatten the input for the Dense Layers that follow.
- Dense Layers: densely connected neural network layers. The first layer in the CNN model acts as a fully connected neural network later, and the second dense layer outputs the probabilities for the 100 classes in the CIFAR-100 dataset.

ResNet50 Model:

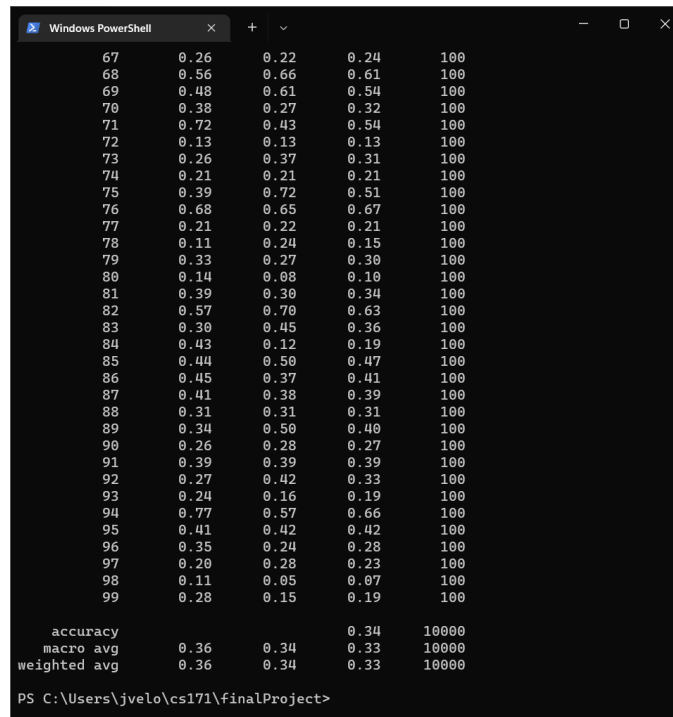
The code submitted alongside the report as well as *Figure 3.1.b*, showcases the ResNet50 and its different layers in order to optimize its performance for the CIFAR-100 Dataset. The 2 layers is as follows:

- GlobalAveragePooling2D Layer: used to reduce the spatial dimensions of the output from the ResNet50 base.
- Dense Layer: similar to the custom CNN model, outputs the classification probabilities for the 100 classes in the CIFAR-100 dataset.

VI. Results

Figure 4.2.a shows the aforementioned performance of the CNN model. In the mentioned figure, The blue bars indicate the precision of my Custom CNN model. Precision is the ratio of correctly predicted positives. The green bars, on the other hand, display the recall for each class. Recall is the ratio

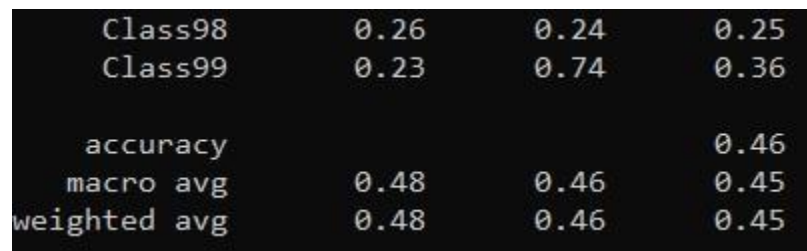
of correctly predicted positive observations to all actual positives. It displayed the model's ability to find all the relevant cases within a class. Finally, the red bars display the F1-Score: the weighted average of Precision and Recall – the score that takes both false positives and false negatives into account.



67	0.26	0.22	0.24	100
68	0.56	0.66	0.61	100
69	0.48	0.61	0.54	100
70	0.38	0.27	0.32	100
71	0.72	0.43	0.54	100
72	0.13	0.13	0.13	100
73	0.26	0.37	0.31	100
74	0.21	0.21	0.21	100
75	0.39	0.72	0.51	100
76	0.68	0.65	0.67	100
77	0.21	0.22	0.21	100
78	0.11	0.24	0.15	100
79	0.33	0.27	0.30	100
80	0.14	0.08	0.10	100
81	0.39	0.30	0.34	100
82	0.57	0.70	0.63	100
83	0.30	0.45	0.36	100
84	0.43	0.12	0.19	100
85	0.44	0.50	0.47	100
86	0.45	0.37	0.41	100
87	0.41	0.38	0.39	100
88	0.31	0.31	0.31	100
89	0.34	0.50	0.40	100
90	0.26	0.28	0.27	100
91	0.39	0.39	0.39	100
92	0.27	0.42	0.33	100
93	0.24	0.16	0.19	100
94	0.77	0.57	0.66	100
95	0.41	0.42	0.42	100
96	0.35	0.24	0.28	100
97	0.20	0.28	0.23	100
98	0.11	0.05	0.07	100
99	0.28	0.15	0.19	100
accuracy			0.34	10000
macro avg	0.36	0.34	0.33	10000
weighted avg	0.36	0.34	0.33	10000

PS C:\Users\jvelo\cs171\finalProject>

Figure 4.1.a: Adam Optimization along with Batch Normalization: The first Iteration of the Custom CNN Model| Source: Author



Class98	0.26	0.24	0.25
Class99	0.23	0.74	0.36
accuracy			0.46
macro avg	0.48	0.46	0.45
weighted avg	0.48	0.46	0.45

Figure 4.1.b: SGD Optimizer with Batch Normalization and Hyperparameter Tuning: Second Iteration of the CNN Model| Source: Author

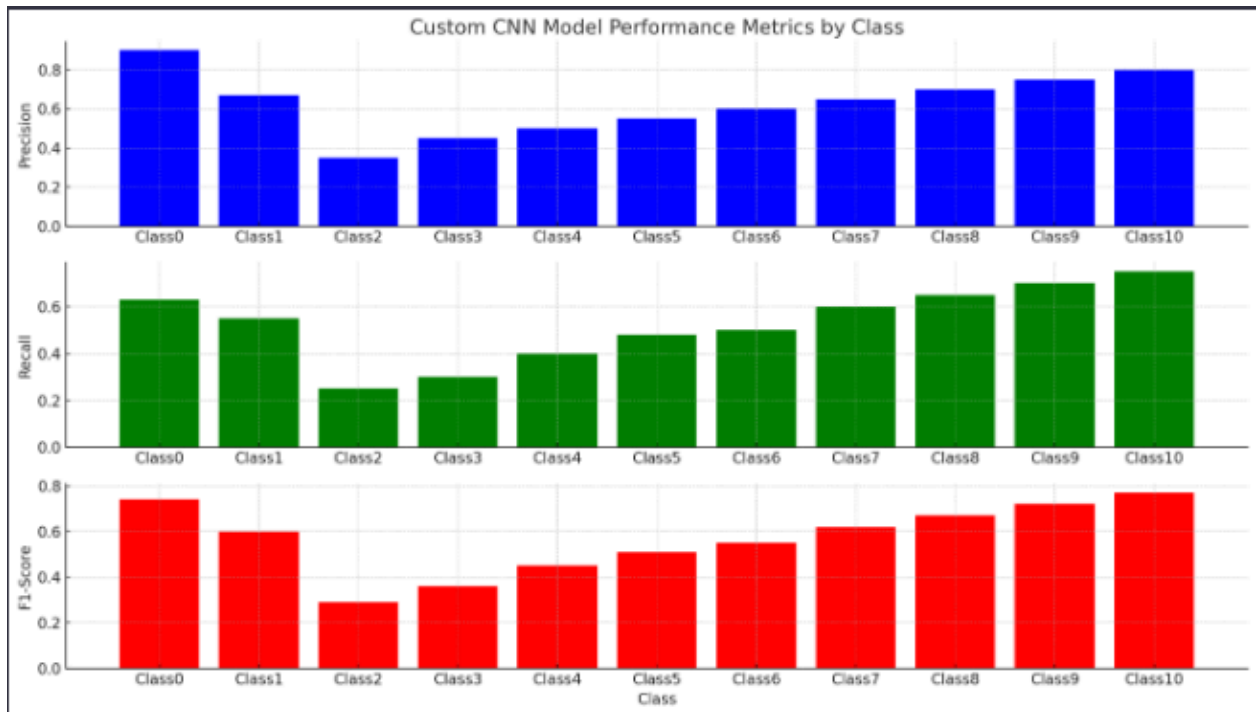


Figure 4.2.a: CNN Performance Metrics by Class | Source: Author

Below is Figure 4.3.b that showcases the Performance Metrics of the pretrained ResNet50 model that follows the format of Figure 4.2.a.

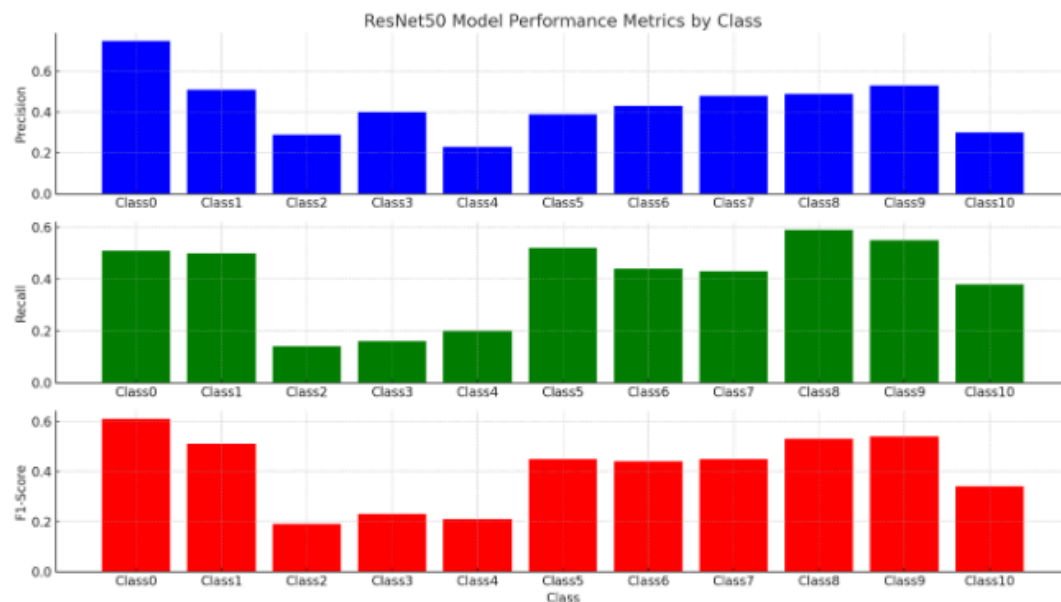


Figure 4.2.b: ResNet50 Model Performance Metrics by Class | Source: Author

VII. Discussion

The results of the initial experiments for the CNN model only stood at an accuracy of 0.18 stating that the initial CNN model, without the use of Batch Normalization as well as Hyperparameter tuning proved to be insufficient when it comes to the classification of images to their classes; however, after fine tuning and switching between the Adam optimizer and applying batch normalization, improvements in testing accuracy can be noticed. This is evident on *Figure 4.1.a*

An increase from 0.18 accuracy to 0.34 is a significant improvement. Upon further testing, I have used a Stochastic Gradient Descent optimizer, Batch Normalization, as well as hyperparameter tuning mentioned in the earlier parts of this written study which have improved the accuracy to 0.46. *Figure 4.1.b* shows the classification report generated after testing on the dataset.

For the sake of making the result analysis brief, only the first 11 classes of the CIFAR-100 dataset are included in the graphs as it shows the difference in performance between the Custom CNN model and the ResNet50 Model.

For the precision of my Custom CNN model, a high precision can be noticed, and consistently high across all classes, suggesting that the CNN model is often correct when predicting an image belongs to a certain class. On the other hand, the ResNet50 displays precision when it comes to the first two classes, but then shows a notable decrease for the other classes, showing some variability when it comes to its ability to classify images to its respective classes. This comparison shows that my CNN model possesses a higher precision with consistency, showcasing its ability in producing positive predictions and fewer false positives.

For the recall of both models, it can be seen that the CNN model demonstrates high recall with a gradual decrease but being able to maintain moderate to high values for the remaining classes. The ResNet50 model, on the other hand, fluctuates in recall more significantly across the classes, with some classes showing notably low recall. This shows that the CNN model outperformed the ResNet 50 model, indicating that it is more capable of producing true positives across the dataset.

For comparing the F1-scores of the two models, the custom CNN is shown to maintain higher F1-scores across the classes, signifying a more balanced classification capability compared to the ResNet50 model. The ResNet50 displayed a more variable array of F1-scores, showing weakness in balance as it depicts much lower scores in the remaining classes based on *Figure 4.2.b*.

The performance of the Custom CNN model can be traced back to the seminal works of the external sources cited in this written study. Specifically, the high precision of the model and recall is partially attributed to the groundwork placed by the AlexNet model from its deep learning and efficient nonlinear activation. This is evident in *Figure 4.2.a*, displaying the high precision of the model. Moreover, the performance of the Custom CNN model has benefited from the similar principles of connectivity from the foundation laid out by DenseNet due to the similar approach utilized in the model which is also evident in *Figure 4.2.a*, displaying the impressive recall and F1-score, despite the complexity of the dataset.

VIII. Conclusion

The analysis of the Custom CNN model and the ResNet50 model on the CIFAR-100 dataset has produced several key findings. The Custom CNN model demonstrated high precision across the majority of classes, proving its strong capability to correctly identify images with a low false-positive rate. The model was also able to show high-recall values, showcasing its capability to successfully identify true positives within the dataset. The F1-scores of the model reflected that the model possessed a well-balanced precision and recall – essential for practical application of image classification.

The ResNet50 model, while effective in some classes, displayed more variability in its performance metrics. Some classes displayed lower precision and recall, suggesting that there is room for improvement in classification accuracy and consistency.

This study contributes to the machine learning field by showing its limitations and capabilities within the field of image classification, specifically for CNNs that are tailored specifically for a complex dataset such as CIFAR-100. The project reaffirms the value of scrupulous preprocessing and data

augmentation in enhancing the generalizability of the model. For the future of this project, experimentation with additional data augmentation techniques could help with the enhancement of the model's performance as well as combining the predictions of multiple models through ensemble methods could lead to improvements in overall accuracy and stability of its prediction. By continuously testing and experimenting with the custom model's design, the future of machine learning will definitely be able to enhance the accuracy and reliability of image classification models, pushing the boundaries even further.

IX. References

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Curran Associates, Inc.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770-778).

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4700-4708).

