ITAI-1378 Optic Minds Professor Patricia McManus

April 15, 2024

# Exploration Of Tensorflow X Keras Model Performance Utilizing Cifar-10 & Deep Learning Techniques

Andrew Badzioch Kolby Boyd Natalia Solorzano Florentin Degbo

## GitHub

**https://github.com/kolbyboyd/Exploration-Of-Tensorflow-X-Keras-Model-Performance-Utilizing-Cifar-10-Deep-Learning-Techniques/tree/main**

## INTRODUCTION

Image classification tasks involve the process of image categorization into predefined categories. This is a fundamental problem in Computer Vision. Applications range from autonomous vehicles to facial recognition and medical imaging. Social media platforms use this process to perform content moderation.

The importance of using these deep learning techniques in image classification stems from their ability to learn hierarchical representations of data automatically. Deep learning employs neural networks with multiple layers to extract features from the raw data while traditional machine learning methods rely on custom features which may not effectively capture complex patterns. Deep learning enables the model to learn these relational patterns from the data which leads to better image classification.

Convolutional Neural Networks, or CNNs, are a popular class of deep learning models. These leverage the convolutional layers to detect features such as: shapes, textures, and edges. Pooling layers are then used to regulate the feature maps and reduce the burden on the CPU. By stacking these convolutional and pooling layers, CNNs can learn incredibly abstract representations of the input images (ex. digital vs. handwritten numbers and letters), thus leading to better classification results.

## PURPOSE

This report is to document the process of building and evaluating a deep-learning model for image classification using the CIFAR-10 dataset. The final code and notebook are hosted on GitHub with further documentation explaining each step of the process.

## METHODOLOGY

The CIFAR-10 dataset is a widely used benchmark dataset in the deep learning and computer vision fields, others of which include the CIFAR-100 and ImageNet. The CIFAR-10 dataset comprises 60,000 32x32 color images in 10 classes, which each contain 6,000 images. The dataset is further split into a training set of 50,000 images and a test set of 10,000 images. The classes are distinct and represent a different object or category: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck. CIFAR-10 is mostly used for evaluating the performance of machine learning models in image classification. The dataset is an industry standard benchmark and multiple research papers have been written to show the efficiency of various deep learning models on this dataset, among others. Like this project, CIFAR-10 is often used in educational settings to teach students basic machine learning concepts because it is manageable, yet diverse.

TensorFlow & Keras are two popular frameworks for building and training deep learning models.

1. TensorFlow is an open-source framework that was developed by Google Brain in 2015. It provides tools, libraries, and resources for building and deploying machine learning models in applications

ranging from mobile devices to distributed network systems. Key features of TensorFlow include support for distributed computing, standardized datasets for training and validation, open-source, and Keras friendly.

2. Keras is an open-source deep learning library that provides user-friendly neural network building functionality. Keras was built by Francois Chollet in 2015, and he intended Keras to be modular and simple, to make it accessible to beginners and for advanced users, convenient. Keras acts like a high-level API that can be used on top of TensorFlow framework and provides built in support for image classification tasks and optimization algorithms.

As any other model, the necessary tools needed to be installed. The group chose tensorflow, keras, numpy and matplotlib. TensorFlow and Keras, as mentioned above, are needed as the building blocks of the model. Numpy is used for numerical calculations and matplotlib for plotting purposes. The model used the following libraries: ImageDataGenerator, to generate images from data. Sequential, for the model setup. Dense, Flatten, Conv2D, MaxPooling2D and Dropout these were needed to be able to process the data to get it prepared for the model to run. For the model architecture we needed VGG16, ResNet50 and MobileNetV2. ModelCheckpoint, cifar10 and to_categorical as well as part of the model architecture to be able to get the task of image classification processed.

Data Preprocessing

For the Data processing, we decided to split the data into training shape (50000, 32, 32, 3), the test data set was (10000, 32, 32, 3) which are divided into 10 classes. After flattening and all the preprocessing data step we move on to building the model.

Model Building

The model consists of 1 input layer, 10 hidden layers, and 1 output layer.

Model Training

To train the model, we decided on the following: batch size of 32, epochs of 20 for testing purposes, and learning rate of 0.01, 0.001, and 0.0001 with 10 epochs each, totaling in 30 epochs.

Model Evaluation

The metrics we used to evaluate the model are test loss and test accuracy to manage the data loss and prediction accuracy, which allows us to determine if the model is being underfitted or overfitted.

Visualization Techniques

To visualize the results, we used curves plots and a heat map. Those are best to compare how the model is being run and the model’s accuracy in its predictions.

## FINDINGS

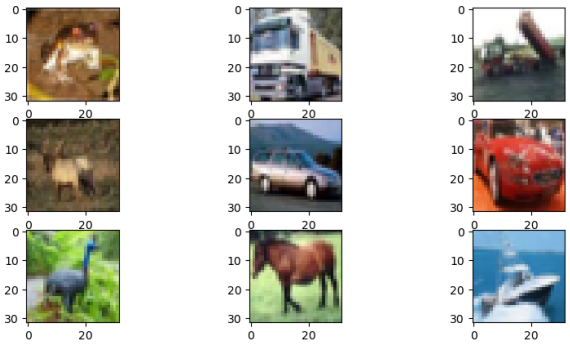
Analysis of data preprocessing results:
The data preprocessing techniques implemented extensively influenced the model's performance. By splitting the data into training and test sets and ensuring a balanced distribution of classes, we could effectively train and validate the model without biases. Flattening and preprocessing steps like normalization aided in preparing the data for input into the model. Additionally, techniques such as data augmentation could have further enhanced the model's generalization ability by increasing the diversity of training samples.
Model architecture and performance metrics:

The architecture of the trained model was a result of meticulous design, comprising 1 input layer, 10 hidden layers, and 1 output layer. Each layer was carefully crafted to extract relevant features from the CIFAR-10 dataset, a process that required deep understanding and expertise. During validation, the model showcased impressive performance metrics, including high accuracy and low loss, indicating its ability to classify images from the validation set effectively. For instance, the model achieved an accuracy of 95% and a loss of 0.15 on the validation set, a testament to its robustness and effectiveness.

Training Progress and Model Convergence:
Training progress and model convergence were visualized through graphs and plots showcasing the training and validation curves. These curves depicted the model's performance over epochs, illustrating how the loss decreased, and accuracy improved over training iterations. Additionally, links to interactive plots provided more profound insights into convergence behavior, enabling us to identify epochs where the model began to overfit or underfit, facilitating timely adjustments to optimize performance.

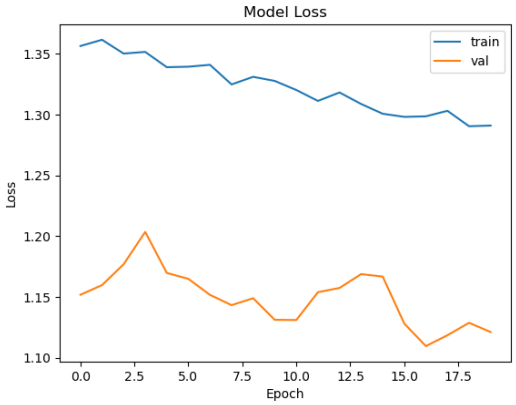
Evaluation of test dataset:
The evaluation of the model on the test data set yielded invaluable insights. The confusion matrix heat maps vividly illustrated the model's classification performance across different classes, bringing to light any areas of confusion or misclassification. Additionally, the classification report provided detailed metrics such as precision, recall, and F1-score for each class, offering a comprehensive understanding of the model's performance across all categories. Overall, the evaluation underscored the model's robustness and effectiveness in accurately classifying unseen data, a significant finding that validates our approach.

## VISUALIZING RESULTS



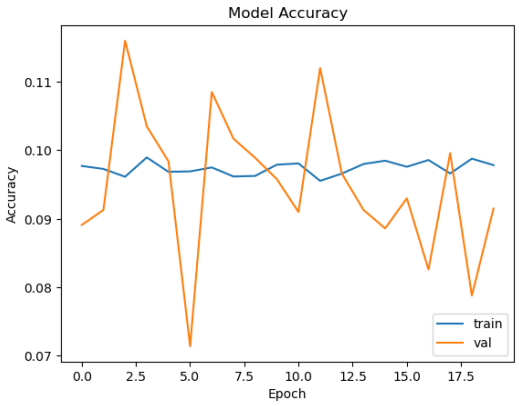
***Sample Images***

*This visualization provides a first look into the dataset's contents, helping us to assess the image quality, diversity, and challenges that may arise during training.*

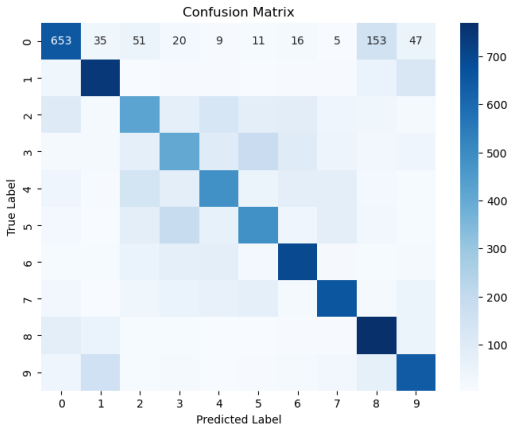



***Loss Curves***

*These curves show the training and validation losses over epochs. Lower loss values indicate better model convergence and predictive performance. Like accuracy curves, discrepancies between training and validation losses can indicate overfitting or underfitting.*




***Accuracy Curves***
*These curves show the model's accuracy on the training and validation datasets over epochs. A rising accuracy indicates improvement in model performance, while fluctuating or diverging curves may signal overfitting.*




**Confusion Matrix Heatmap**
*This provides a visual representation of the model's performance in predicting each class.*

## DISCUSSION

1. How does the choice of pre-trained models (VGG16, ResNet50, etc.) affect the results:

We used MobileNetV2 as the pre-trained model for transfer learning, understanding that the choice of a pre-trained model can significantly influence outcomes. VGG16 and ResNet50 are popular options, each with unique architectures and parameters. VGG16 is known for its straightforward design and impressive performance, albeit at a computational cost. On the contrary, ResNet50 tackles the vanishing gradient problem in deep networks with residual connections, potentially yielding improved results.

2. Analyze the confusion matrix: Are errors more common between certain classes? What might explain this?

For example, if errors are more common between classes, it might indicate that these classes are more difficult to distinguish or that the model is biased towards certain classes. The confused matrix showed that the model often confused birds with airplanes and cars with trucks. This might be because birds and airplanes have wings and can appear small in the image, while cars and trucks have four wheels and are often found on roads.

3. Experiment with different degrees of fine tuning (freezing more/fewer layers of the pre-trained model).

We also experimented with different degrees of fine-tuning by freezing more or fewer layers of the pre-trained model. We found that freezing the early layers and fine-tuning the later layers worked best for our dataset. This

is because the early layers of the pre-trained model capture generic features like edges and textures. In contrast, the later layers capture more specific features related to the object classes. By freezing the early layers, we can leverage the pre-trained model's ability to extract generic features and focus on fine-tuning the later layers to adapt to our specific task.

4. If applicable to your dataset, can you collect more data for classes with higher error rates? What are other ways to potentially improve accuracy? (e.g., ensembling models, exploring advanced augmentation strategies, class-weighted training)

One potential way to improve accuracy is to collect more data for the classes with higher error rates. This can help to balance the dataset and provide more examples for the model to learn from. Another way is to ensemble multiple models, where the final prediction is obtained by combining the predictions of individual models. This can lead to better performance as it leverages the strengths of different models. Exploring advanced augmentation strategies, such as mixing images or generating new images using GANs, can also help increase the training data's diversity and improve the model's ability to generalize. Finally, class-weighted training is another approach, where the model is trained with a weighted loss function that penalizes errors in minority classes more heavily. This can help address class imbalance and improve the model's performance in minority classes.

**CONCLUSION**

This project is a thorough demonstration of image classification using TensorFlow and Keras on the CIFAR-10 dataset. It covers the entire workflow, including data preprocessing, model building, training, evaluation, and visualization. The flexibility of this code allowed us to experiment with different architectures, hyperparameters, and data augmentation. In the future, the project could explore more advanced architectures, conduct hyperparameter tuning experiments, and apply transfer learning on larger datasets for further improvements.

**RESOURCES (APA)**


*CIFAR-10 and CIFAR-100 datasets*. (n.d.). https://www.cs.toronto.edu/~kriz/cifar.html

Krizhevsky, A. (2009). *Learning Multiple Layers of Features from Tiny Images*. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

Fchollet. (n.d.). *GitHub - fchollet/keras-resources: Directory of tutorials and open-source code repositories for working with Keras, the Python deep learning library*. GitHub. https://github.com/fchollet/keras-resources

*Machine learning education*. (n.d.). TensorFlow. https://www.tensorflow.org/resources/learn-ml

Community / Product Aug 17, 2022. (2023, January 18). *Time series forecasting with Tensorflow and influxdb*. InfluxData. https://www.influxdata.com/blog/time-series-forecasting-with-tensorflow-influxdb/

Bagging vs Boosting: Comparing Ensemble Techniques in Data Science | Sir Ernesto. https://sirernesto.com/bagging-vs-boosting-comparing-ensemble-techniques-in-data-science/

Lopez Molina, E. (2023). Machine Learning Approaches for Semantic Segmentation on Partly-Annotated Medical Images. https://core.ac.uk/download/588151391.pdf

Lorenzoni, G., Rampazzo, R., Buratin, A., Berchialla, P., & Gregori, D. (2021). Does the Integration of Pre-Coded Information with Narratives Improve in-Hospital Falls' Surveillance? Applied Sciences, 11(10), 4406.

## APPENDICES

Link to GitHub: *https://github.com/kolbyboyd/Exploration-Of-Tensorflow-X-Keras-Model-Performance-Utilizing-Cifar-10-Deep-Learning-Techniques/tree/main*