Image Classification With CNN

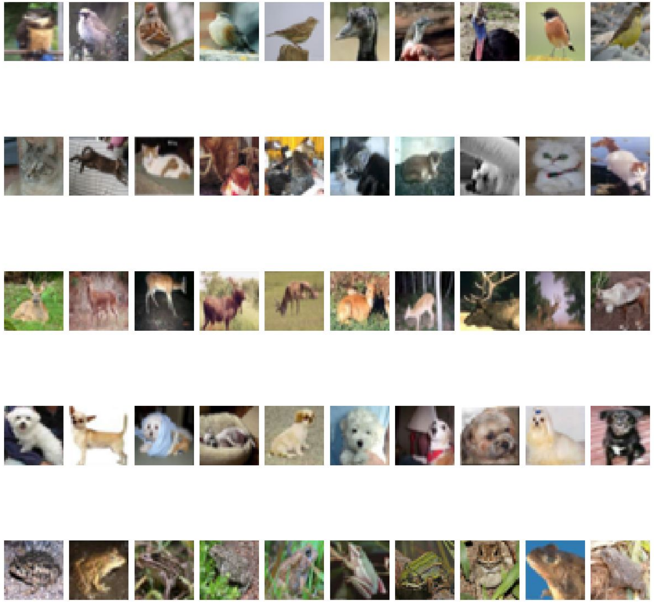
Group - 5

Ahmad - Georg - Matheus - Sofia

Methods (preprocessing) ⚙️

- **Dataset Used:** CIFAR-10
- Reviewed all dataset classes and organized them into training vectors.
- Converted class labels into **one-hot encoding** for model compatibility.
- Filtered the dataset to include only animal classes
- Remapped class indexes accordingly
- Model Evaluation: Training on animal classes showed **no significant overfitting** in our experiments.

Sample of the selected classes



Model Evaluation 📊

- Trained models on the animal subset.
- Observed **no significant overfitting** in experiments
- Evaluated multiple architectures to find best performance

Experiments Overview 🔬

We tested multiple CNN-based approaches:

- Simple CNN
- MobileNetV2 (Transfer Learning)
- CNN with Data Augmentation
- ResNet

Each experiment aimed to improve model accuracy and reduce loss.

Image Classification - Simple CNN 📷🔍

Configuration: 10 epochs, batch size 64, Adam optimizer, ReLU & Softmax activation, 4 neuron layers.

Results:

- ❏ Training Accuracy: 0.6849 | Loss: 8.905
- ❏ Validation Accuracy: 0.6835 | Loss: 9.255

Observation: Basic CNN performed reasonably but showed high loss, suggesting room for improvement (possible main causes: underfitting or lack of model depth)

Metrics - Parameters Example:

Hardware / Runtime		Colab GPU (T4)	CPU (TensorFlow oneDNN optimizations, ~8–10 s per epoch → ~4–5 min total)	Colab GPU T4
Activation functions		ReLU / Softmax	ReLU (hidden layers), Softmax (output layer)	Adam
Optimizer	Defined during model compilation (model.compile)	Adam	Adam	adam_opt
Dataset	Defined at data loading (cifar10.load_data())	CIFAR-10	CIFAR-10 (10 classes, 32×32 color images)	
Learning Rate (LR)	Inside optimizer setup (Adam(learning_rate=0.001))	1	1	default 0,001
Epochs	Training configuration (model.fit(...))	10	30	10
Batch Size	Training configuration (model.fit(...))	64	64	128
Training Time (min)		around 10 min	4–5 minutes total	30
Loss Function	Defined in model.compile(loss=...)	categorical_crossentropy	Categorical Crossentropy	categorical_crossentropy
Final Loss	Last epoch in training log	8.905	~0.64 (train), ~0.87 (val/test)	0,9642
Final Accuracy (Train)	Last epoch in training log	0.6849 (≈68%)	~0.76	0,6431
Validation Loss	Last epoch in training log	9.255	0.87	1.0023
Validation Accuracy	Last epoch in training log	0.6835 (≈68%)	0.71	0,6225
Notes / Comments		Try more epochs (20–25)	Good baseline CNN. Improving with data augmentation, dropout tuning, or learning rate scheduling may raise accuracy above 75%	I used the simple model so I can understand how is done

Image Classification - VGG-Style CNN

📷🧱

Configuration:
45 epochs, batch size 512, Adam optimizer (learning rate = 0.001), ReLU & Softmax activation, 11 convolutional + dense layers

Results:

- ❑ Training Accuracy: 0.733 | Loss: 1.672

Observation:
The VGG-style CNN showed improved accuracy compared to the simple CNN.
Training was more stable due to batch normalization, but the model still displayed moderate loss — indicating potential for further optimization or regularization.

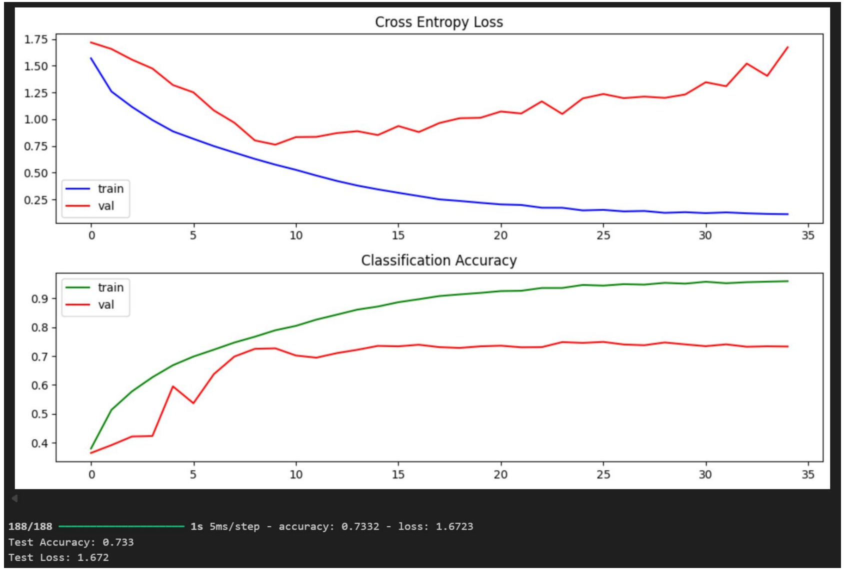


Image Classification - CNN with Data Augmentation - Early Stopping 📷✨

Configuration:
50 epochs, batch size 128, Adam optimizer (learning rate = 0.001), ReLU & Softmax activation, 4 convolutional + dense layers

Results:

- ❏ Training Accuracy: 0.8505 (~85%) | Loss: 0.3791 (38%)
- ❏ Validation Accuracy: 0.6892 (~69%) | Loss: 1.1196

Observation:
Data augmentation helped slightly improve validation performance, and early stopping prevented overfitting.

Image Classification with Resnet 📷📈

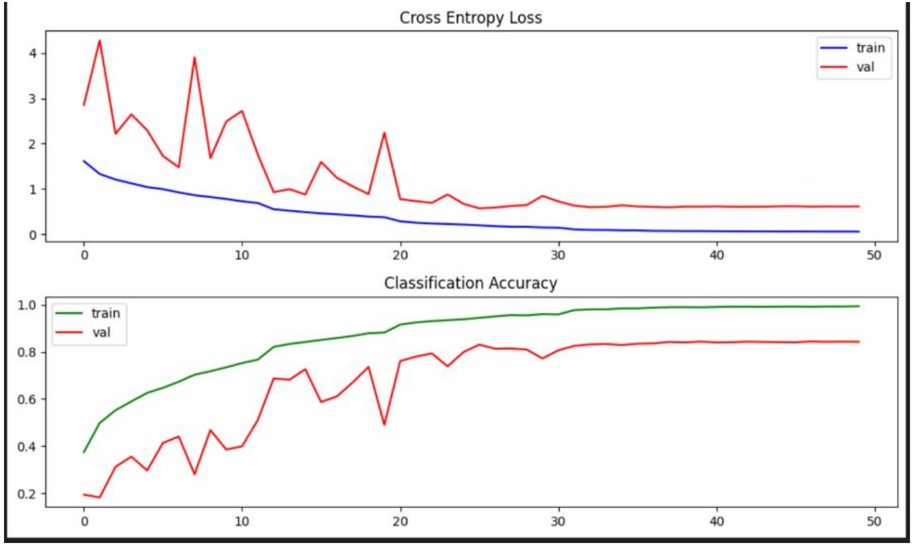
Core Concept

- **Addressing Vanishing Gradient**: Allows models to learn identity functions.
- **Forward Original Inputs**: Enables unchanged re-passing of inputs.

Benefits

- **Simplified Learning**: Easier adjustments when function complexity is unnecessary.
- **Stable Gradient**: Enhances training efficiency in deeper networks.
- **High Efficiency and Accuracy**: Facilitates effective model construction

Resnet Model 📈



- **Optimizer**: **Adam** optimizer is used with the default learning rate of 0.001.
- **Loss Function**: The model uses **categorical crossentropy** to compute the loss.
- **Learning Rate Scheduler**: The learning rate is reduced by a factor of 0.4 if validation loss does not improve over 5 epochs.

```
188/188 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step - accuracy: 0.8425 - loss: 0.6126
Test Accuracy: 0.842
Test Loss: 0.613
Recall: 0.843
F1 Score: 0.843
```

Image Classification with Transfer Learning 🧠🔄

Configuration:
10 epochs, batch size 128, Adam optimizer (learning rate = 0.001), ReLU & Softmax activation, 4 convolutional + dense layers

Results:

- Training Accuracy: 0.6431 | Loss: 0.9642
- Validation Accuracy: 0.6225 | Loss: 1.0023

Observation:
Using transfer learning allowed the model to leverage pre-trained features, which improved performance and reduced training time. This approach is particularly useful when working with limited data, and it showed good validation accuracy while learning efficiently.

Winning Model 🏆

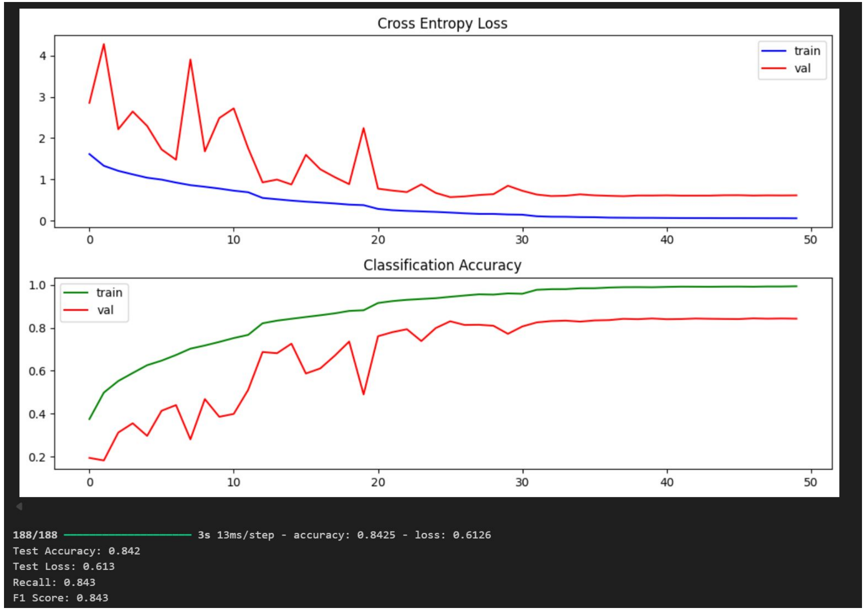
Best Model: ResNet
Accuracy: 0.8425 | **Loss:** 0.6126

Quick recap of alternatives:

- ❑ Simple CNN
- ❑ MobileNetV2 (Transfer Learning)
- ❑ CNN with Data Augmentation

Key Observations:

- ❑ ResNet outperformed all other models in both accuracy and stability.
- ❑ Adding **batch normalization** in simpler CNNs improved training stability and reduced loss, highlighting the importance of normalization in deep learning.



Takeaways 🔑

- Residual connections in ResNet improve learning for deeper networks.
- Outperformed simple CNN and augmented CNN.

Recap / Conclusions

- **ResNet** is the most effective model for CIFAR-10 classification.
- **Data augmentation** improves generalization but alone is insufficient.
- **Batch normalization** in simple CNN boosts performance.
- **Transfer learning**: no good results on the dataset

Challenges

- Overfitting when training on limited subsets of data.
- High loss with simple CNN, even with small architectures.
- Choosing optimal hyperparameters (batch size, learning rate, epochs).

Key Learnings

- Preprocessing (one-hot encoding, normalization) is crucial for CNNs.
- Data augmentation mitigates overfitting but doesn’t fully replace model complexity.
- Deep architectures like ResNet handle vanishing gradient and outperform shallow networks.
- Combining batch normalization and advanced architectures yields the best results.

Closing Remarks ✍️

Our image classification project demonstrated the **power of deep learning** and the impact of architectural choices on model performance.
Through multiple experiments — from a **simple CNN** to **data augmentation** and **ResNet** — we observed clear improvements in accuracy, stability, and generalization.

The journey showed that:

- Small adjustments like **batch normalization** and **augmentation** can make a significant difference.
- **ResNet**'s residual learning architecture remains one of the most reliable methods for image classification.
- Continuous experimentation and fine-tuning are key to achieving high performance in real-world applications.

Final Thoughts

> "In deep learning, progress comes not only from deeper models —
> but from deeper understanding."