DEEP LEARNING-IMAGE CLASSIFICATION WITH CNN

G3

Aurèle Lambert

Paul Müller

Enrique García

Content

2. Summary

Challenge

Build a CNN with and without transfer learning

Data set = CIFAR-10

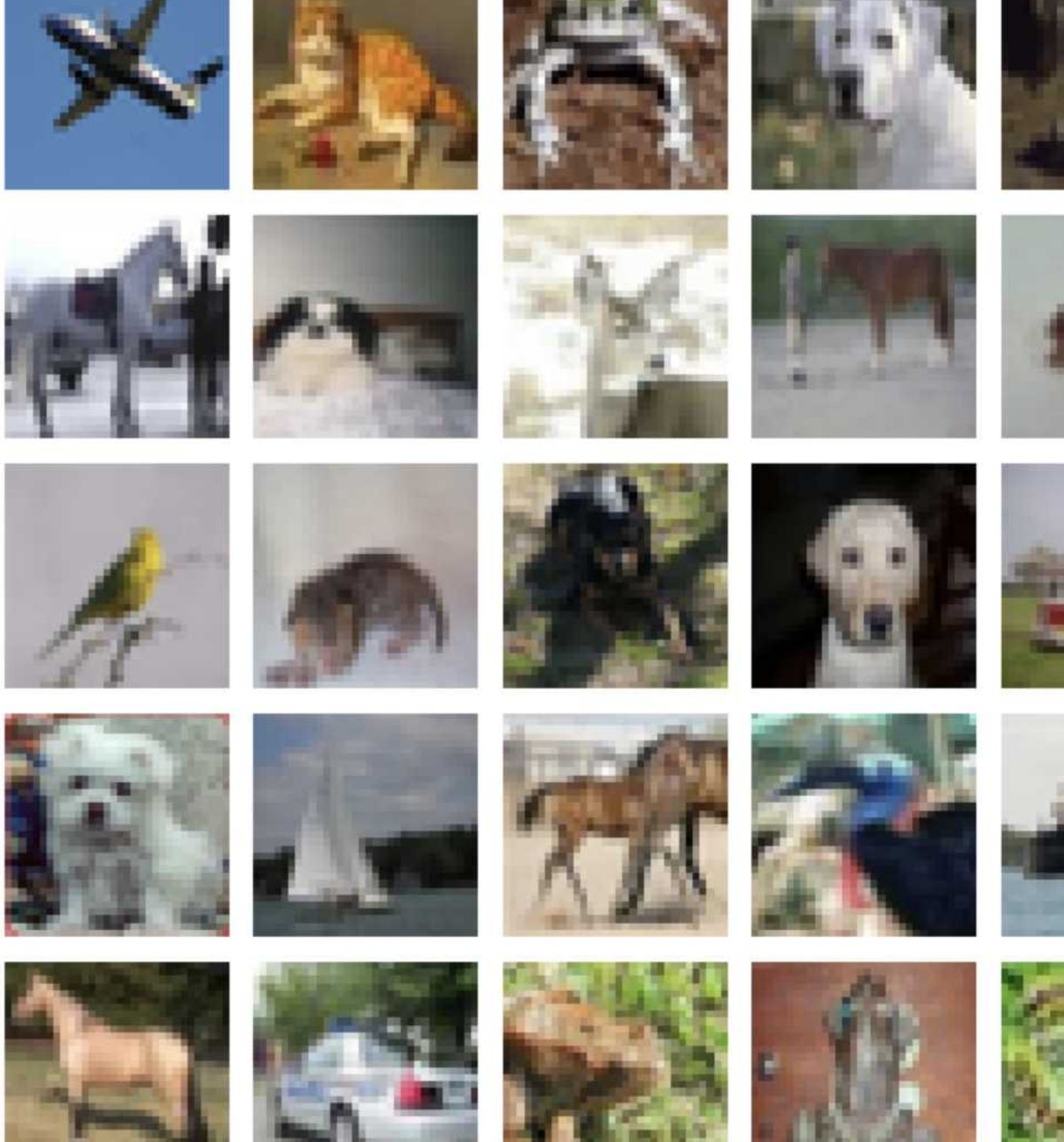
Considered using data with medium images but issues with importing the data set correctly

3 homebrew models developped

- *Top accuracy achieved: 88.10%*

3 transfer models tested

- *Top accuracy achieved: 64.96%*



3. Overview of the problems

Data Cleaning

Discuss the biggest obstacle or mistake you encountered during this project.

- Failed to import and use Kaggle medium image data → moved to CIFAR
- Scaling the data to 64 × 64
- Crashed computer when upscaling to 244×244

Model Designing

Share what you learned from it and how it influenced your project.

- Finding a plateau at 65 % and later at 87 % accuracy, screwing up by optimization
- Model slowed down, with more and more improvements, time and resource consuming

Transfer Learning

Is there anything you would do differently in hindsight?

- Effort to find the right model: requires to read documentation & understand constraints
- Effort to adapt the dataset to correctly fit pre-learned models
- Complex to optimize pre-learned models - painful in a short amount of time

4. Data Preprocessing

Data CIFAR-10

- *Clean from the get go*
- *Concerned that image size could impact model results:*
  Can human even easily attribute the correct label for some pics?

For homebrew and transferred models

- *Hot encoding for labels*
- *Normalization*
- *Data augmentation*
- *Image resizing (model dependant)*

Issues encountered with data

- *Had trouble importing medium image → switched to CIFAR-10*
- *Session crashed because of 32×32 → 224×224 transformation*

5. Designed CNN Architecture

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 64, 64, 32)	896
batch_normalization (BatchNormalization)	(None, 64, 64, 32)	128
dropout (Dropout)	(None, 64, 64, 32)	0
conv2d_1 (Conv2D)	(None, 64, 64, 32)	9,248
batch_normalization_1 (BatchNormalization)	(None, 64, 64, 32)	128
max_pooling2d (MaxPooling2D)	(None, 32, 32, 32)	0
conv2d_2 (Conv2D)	(None, 32, 32, 64)	18,496
batch_normalization_2 (BatchNormalization)	(None, 32, 32, 64)	256
dropout_1 (Dropout)	(None, 32, 32, 64)	0
conv2d_3 (Conv2D)	(None, 32, 32, 128)	73,856
batch_normalization_3 (BatchNormalization)	(None, 32, 32, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 128)	0
dropout_2 (Dropout)	(None, 16, 16, 128)	0
conv2d_4 (Conv2D)	(None, 16, 16, 128)	147,584
conv2d_5 (Conv2D)	(None, 16, 16, 128)	147,584
dropout_3 (Dropout)	(None, 16, 16, 128)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 128)	0
dense (Dense)	(None, 128)	16,512
dropout_4 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

4 Convolutional Blocks:

- Conv2D layers with increasing filters (32, 64, 128).
- ReLU activation, Batch Normalization, and Dropout (20%-50%).
- MaxPooling2D for spatial reduction.

Global Average Pooling:

- Replaces Flatten for dimensionality reduction.

Fully Connected Layers:

- Dense (128 units, ReLU, Dropout 50%).
- Output: 10 units, Softmax for classification.

Key Highlights:

- Dropout for regularization.
- MaxPooling for downsampling.
- Total Parameters: Efficient and optimized for performance.

G3 Model result:

	Class	Precision	Recall	F1-Score
0	airplane	0.885344	0.888	0.886670
1	automobile	0.935103	0.951	0.942985
2	bird	0.856100	0.821	0.838183
3	cat	0.809218	0.755	0.781169
4	deer	0.842897	0.896	0.868638
5	dog	0.823472	0.849	0.836041
6	frog	0.878187	0.930	0.903351
7	horse	0.935586	0.886	0.910118
8	ship	0.914751	0.955	0.934442
9	truck	0.944915	0.892	0.917695

G3 Model Architecture:

```
dict_keys(['accuracy', 'loss', 'val_accuracy',
Test loss: 0.3564315438270569
Test accuracy: 0.8823000192642212
```

Test accuracy: 0.88 %

6. Details on the optimization techniques used

Data Augmentation:
- *Increased data diversity with transformations like rotation, zoom, and flips.*

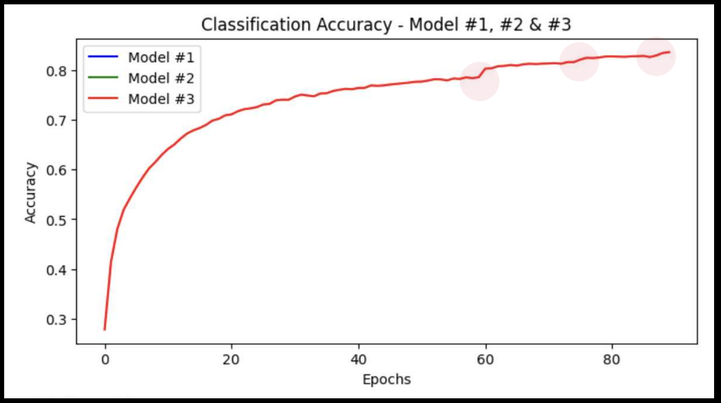
Early Stopping:
- *Halted training after 10 epochs of no validation improvement, restoring best weights.*

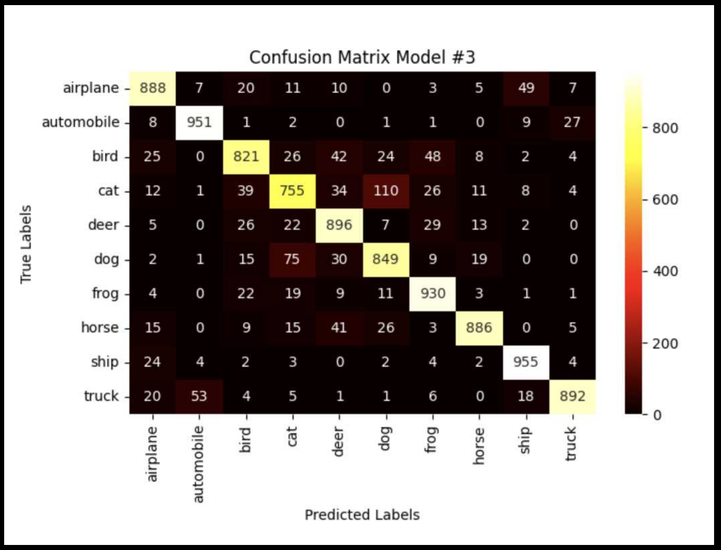
Learning Rate Scheduling:
- *Reduced learning rate dynamically using ReduceLROnPlateau.*

Optimizer:
- *Used AdamW with an initial learning rate of 0.001.*

Batch Size & Epochs:
- *Batch size of 512, epochs set to 150 with early stopping managing training duration.*

Class Weight Adjustment:
- *Addressed class imbalance (e.g., "cat" class) with computed weights (not used due to time constraints). // missing time for strong improvement*





7. Transfer Models

3 transfer models were tested

Selected by reading the Kiras website application documentation pages base on accuracy and processing power required

EfficientNetB1

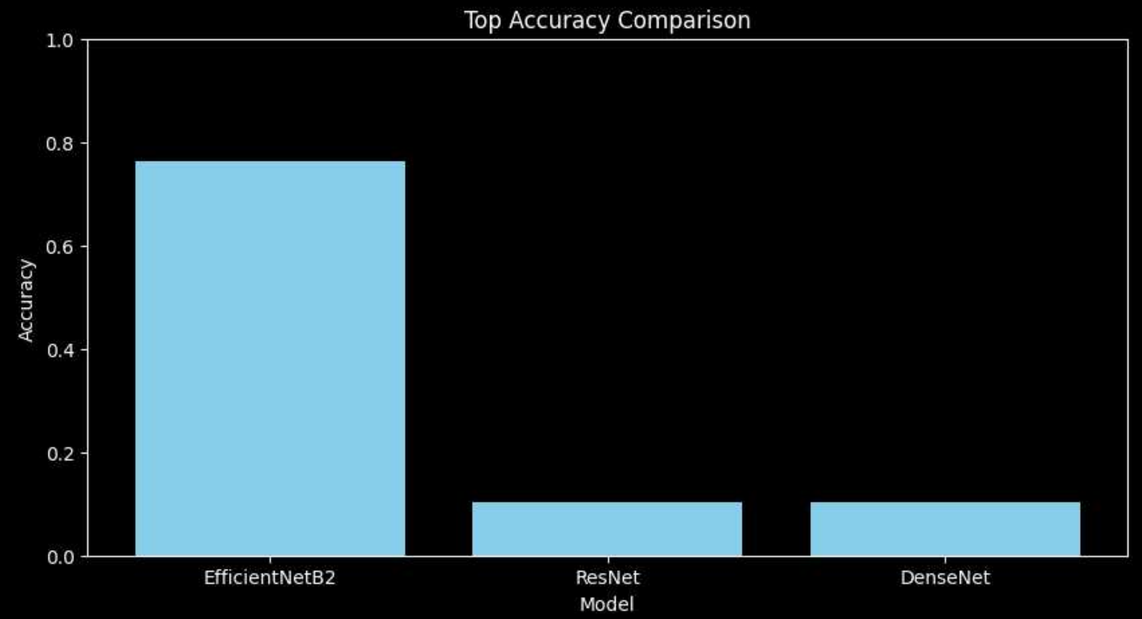
- *Base accuracy (no tweak): 44.19%*
- *Top accuracy achieved: 64.96%*

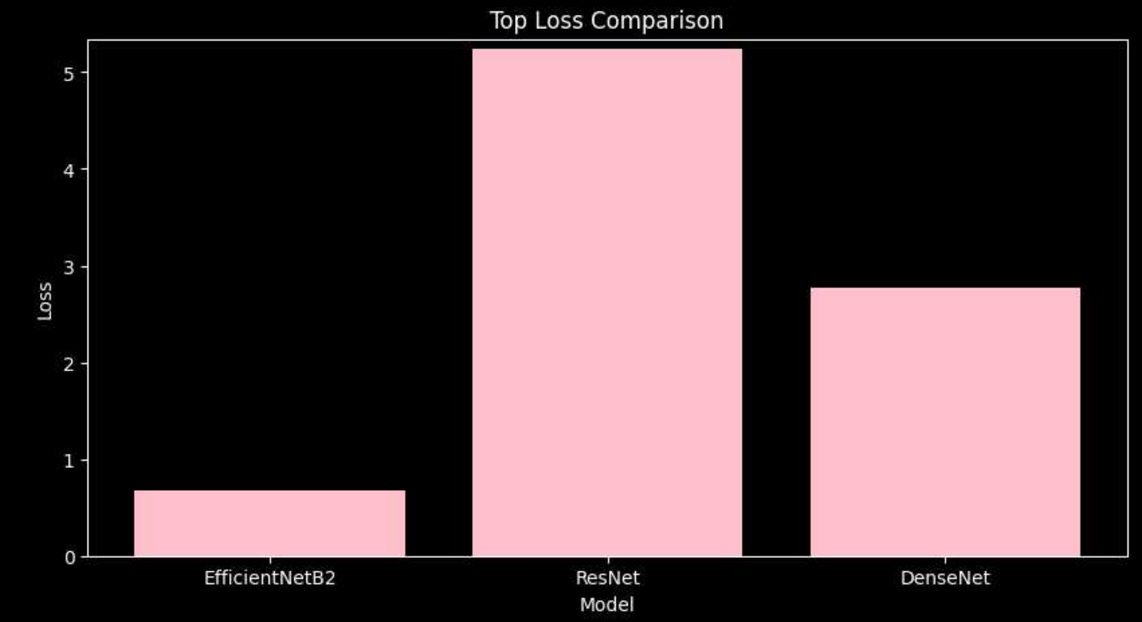
ResNet101V2

- *Base accuracy (no tweak): 7,28%*
- *Top accuracy achieved: 11%*

DenseNet121

- *Base accuracy (no tweak): 10,75%*
- *Only reached around 10% due to resource limitations and insufficient fine-tuning.*





8. Evaluation

Homebrew wins by ~23% accuracy

Homebrew - Acc: 88,1% | Loss: 0.3564
Best transfer - Acc: 64.96% | Loss: 1.40

Homebrew took less time to achieve good performance

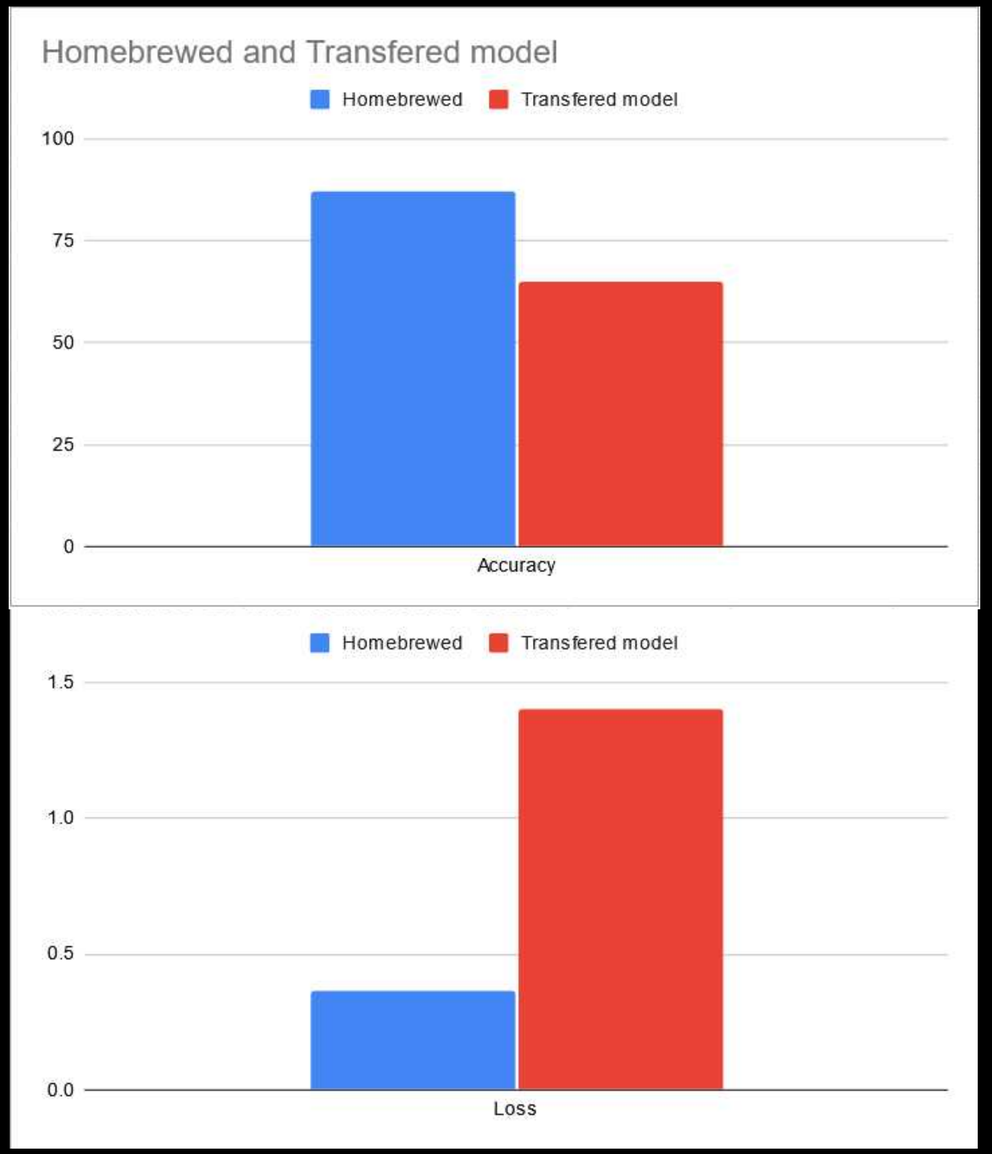
- *>85% accuracy achieved in 1 day*
- *Easier to tweak for improvement*

Transfer learning

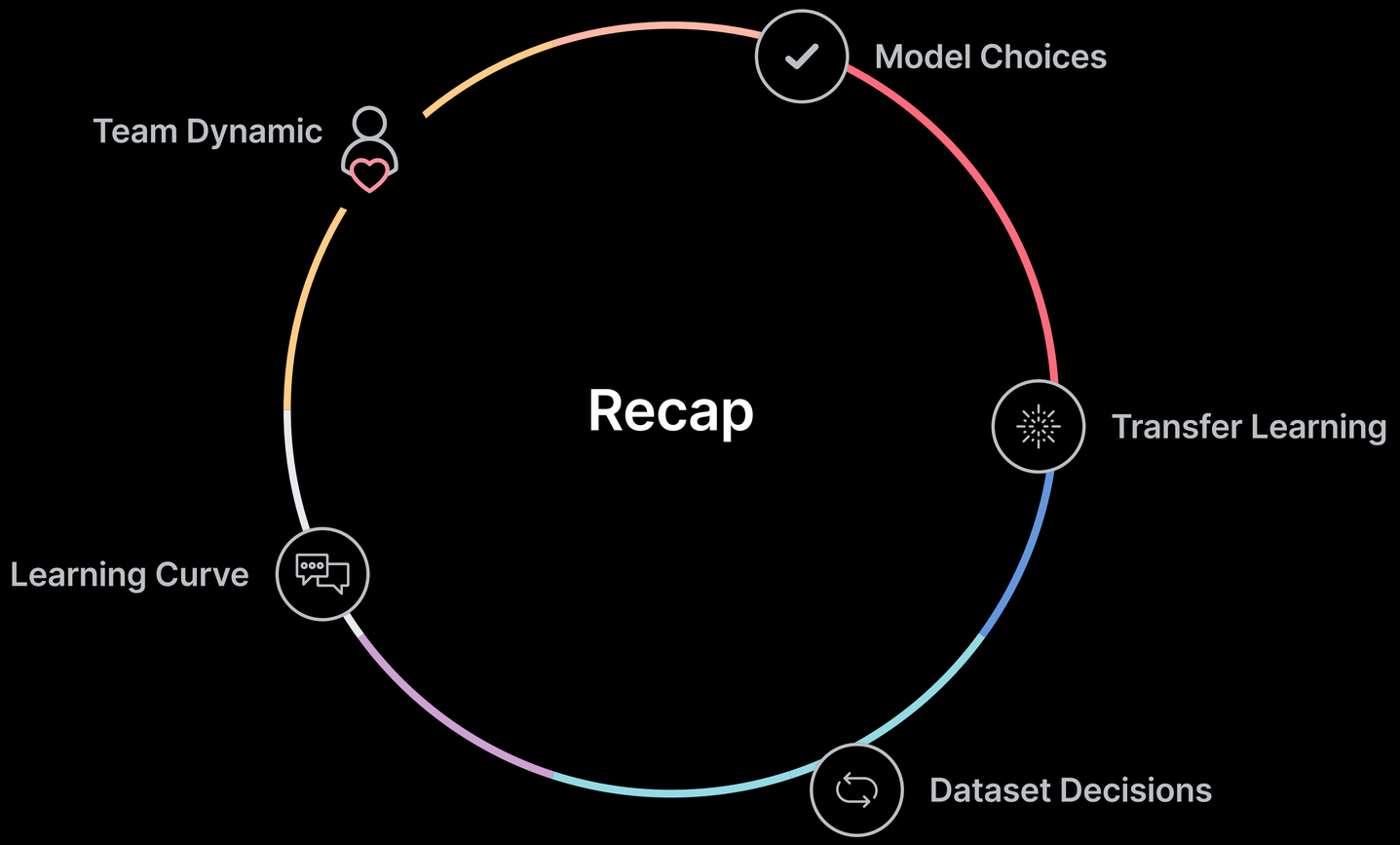
- *Sticked to top performance ~64%*
- *More complex & time consuming to tweak for improvement*

Critic and interrogation

- *Transfer models were trained with higher quality images and may not be adapted to CIFAR 32×32*
- *Found examples on Internet of much better performance with Resnet achieving >80% accuracy on CIFAR*



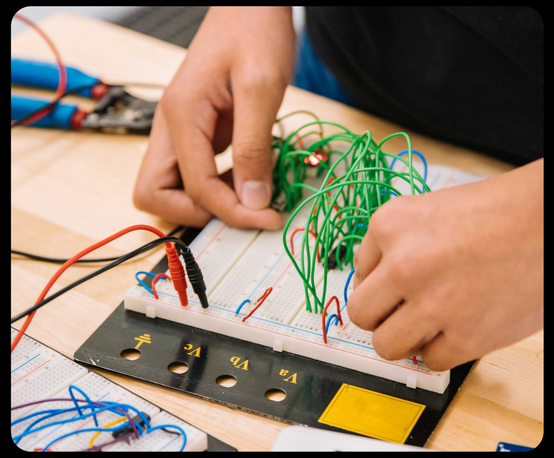
10. Retrospective

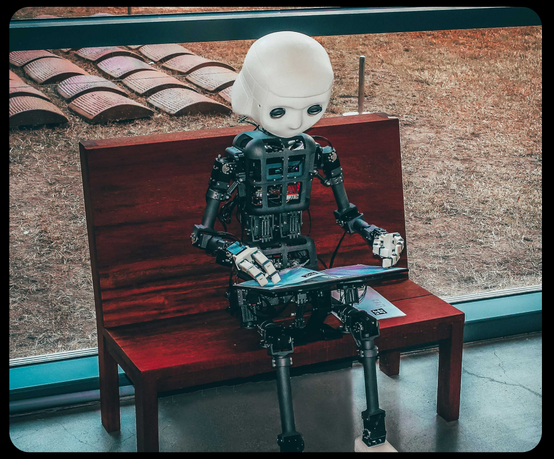


- Group Dynamics
- Model Choices
- Transfer Learning
- Dataset Decisions
- Learning Curve

6. Conclusion and Insights (2-3 slides)

- Hardware Constrains
- Transfer Learning requierements
- Problem Solving
- Data preprocessing
- ChatGPT





Thank you

Time for feedback