CNN Experiments on CINIC-10

PAULINA KULCZYK,

JAN POGŁÓD

BASIC ARCHITECTURES

- **Single Convolution CNNs**: A simple architecture with one convolutional layer, intended to quickly test baseline accuracy and overfitting tendencies.
- **Double Convolution CNNs**: An extended version with two convolutional layers, aimed at capturing more complex spatial features while maintaining computational simplicity.

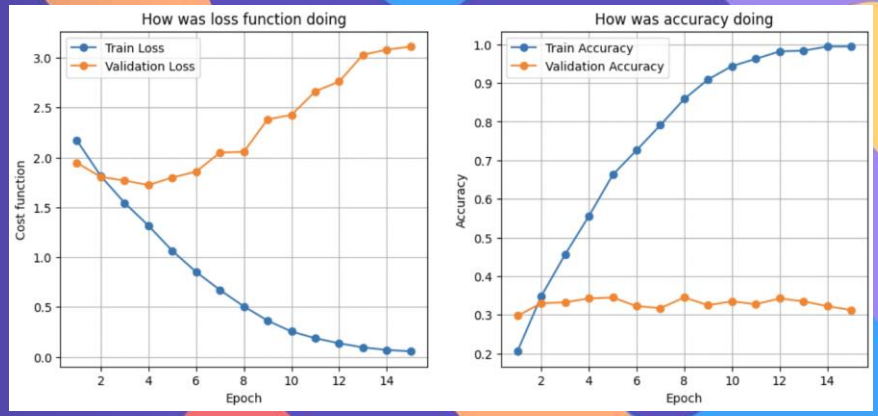
Table 1: Single Convolution CNNs Architecture

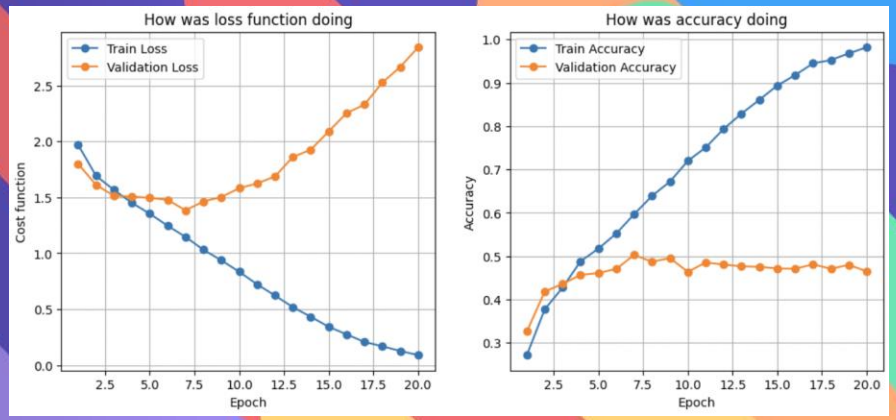
Layer Type	**Parameters**	**Output Shape**
Input	-	(32, 32, 3)
Conv2D	32 filters (3x3), ReLU	(32, 32, 32)
MaxPooling2D	(2, 2) pool size	(16, 16, 64)
Flatten	-	$(16 \cdot 16 \cdot 32)$
Dense	128 neurons, ReLU	(128)
Dense	10 neurons, softmax	(10)

Table 2: Double Convolution CNNs Architecture

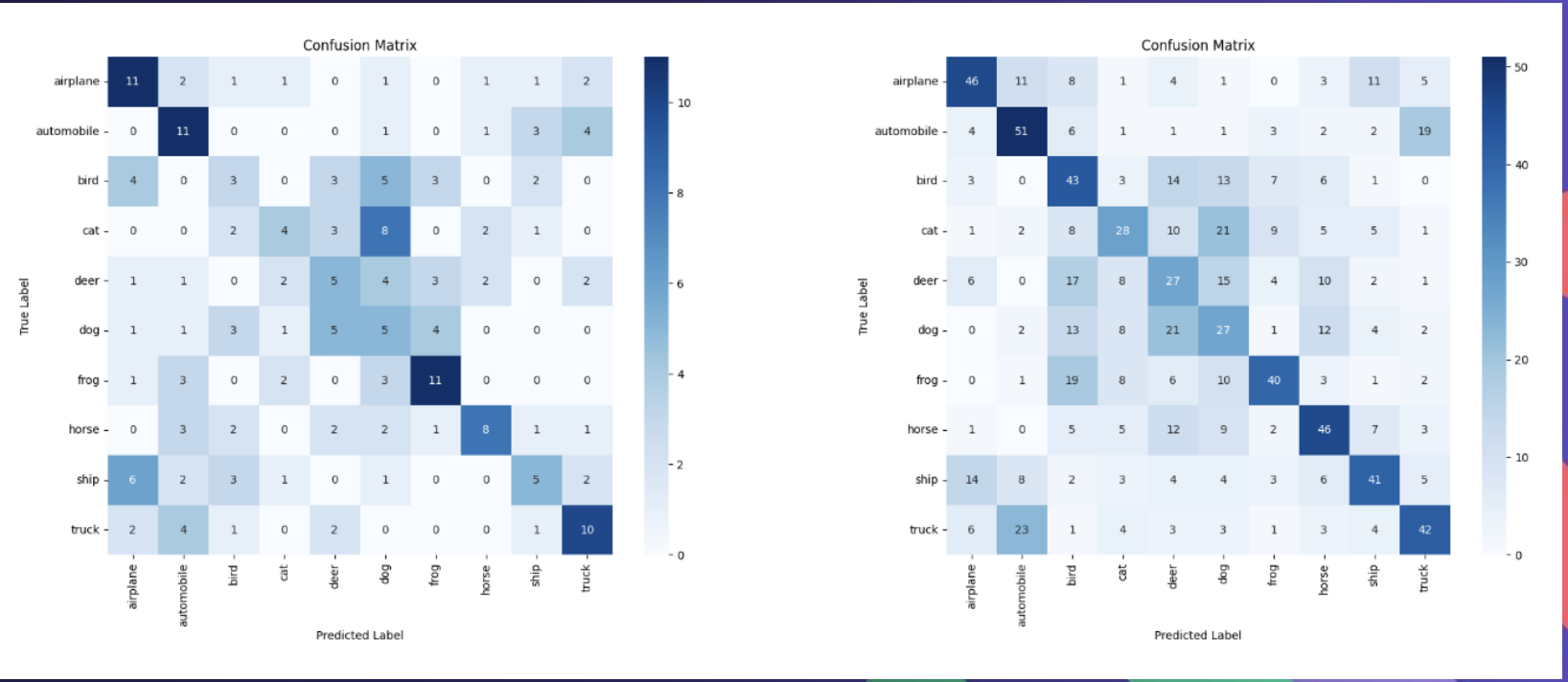
Layer	**Parameters**	**Output Shape**
Input	-	(32, 32, 3)
Conv2D	32 filters (3×3), ReLU, same	(32, 32, 32)
MaxPooling2D	(2, 2) pool size	(16, 16, 32)
Conv2D	64 filters (3×3), ReLU, same	(16, 16, 64)
MaxPooling2D	(2, 2) pool size	(8, 8, 64)
Flatten	-	(4096)
Dense	128 neurons, ReLU	(128)
Dense	10 neurons, softmax	(10)

TRAINING PROCESS

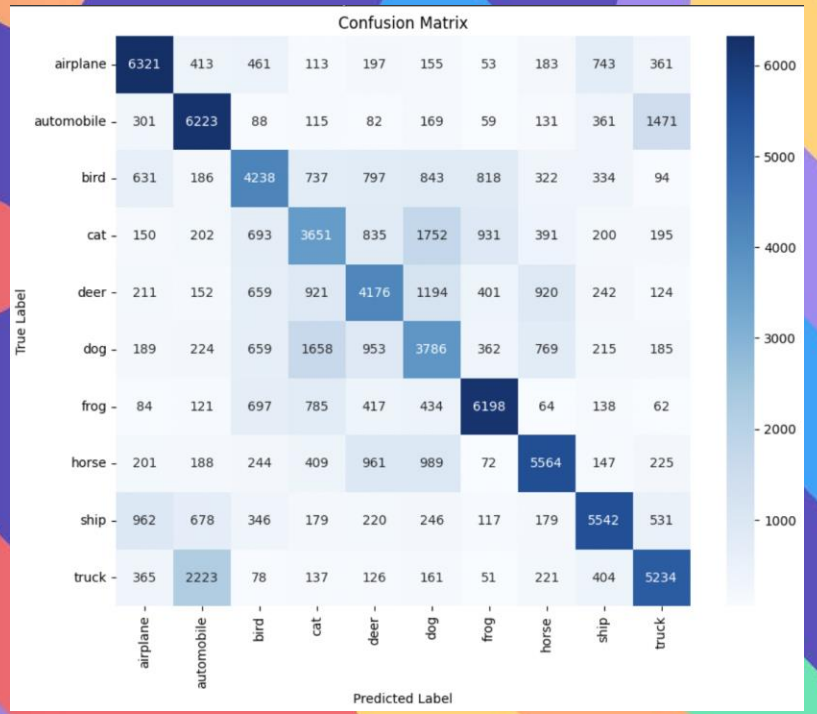




SMALL DATASETS (2000 AND 9000 IMAGES)



FULL DATASET



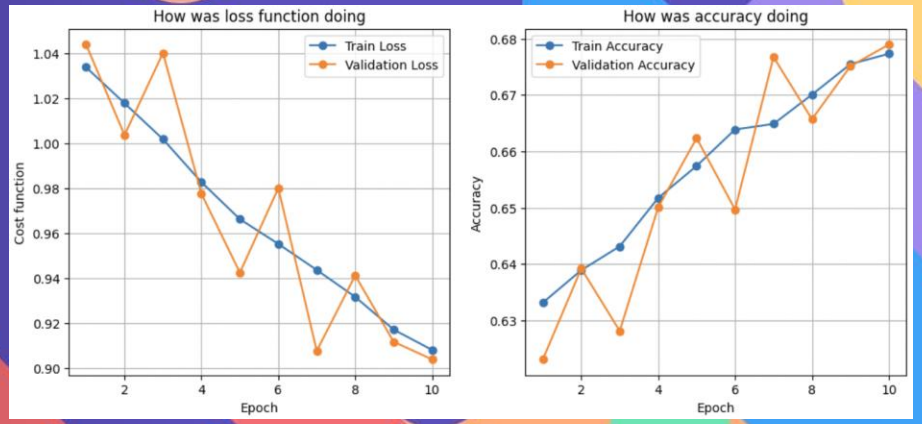
BASIC ARCHITECTURES

Table 3: Deep CNNs Architecture

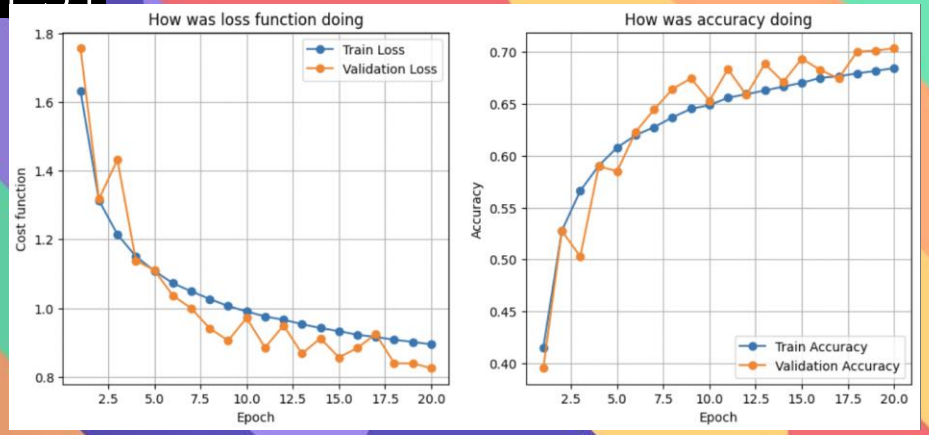
Layer Type	Parameters	Output Shape
Input	-	(32, 32, 3)
Conv2D	32 filters (3x3), ReLU	(32, 32, 32)
BatchNormalization	-	(32, 32, 32)
MaxPooling2D	(2, 2) pool size	(16, 16, 32)
Dropout	0.25 rate	(16, 16, 32)
Conv2D	64 filters (3x3), ReLU	(16, 16, 64)
BatchNormalization	-	(16, 16, 64)
MaxPooling2D	(2, 2) pool size	(8, 8, 64)
Dropout	0.25 rate	(8, 8, 64)
Conv2D	128 filters (3x3), ReLU	(8, 8, 128)
BatchNormalization	-	(8, 8, 128)
MaxPooling2D	(2, 2) pool size	(4, 4, 128)
Dropout	0.25 rate	(4, 4, 128)
Flatten	-	$(4 \cdot 4 \cdot 128)$
Dense	512 neurons, ReLU	(512)
BatchNormalization	-	(512)
Dropout	0.5 rate	(512)
Dense	256 neurons, ReLU	(256)
BatchNormalization	-	(256)
Dropout	0.5 rate	(256)
Dense	10 neurons, softmax	(10)

DEEP CNN WITH REGULARIZATION AND DROPOUT

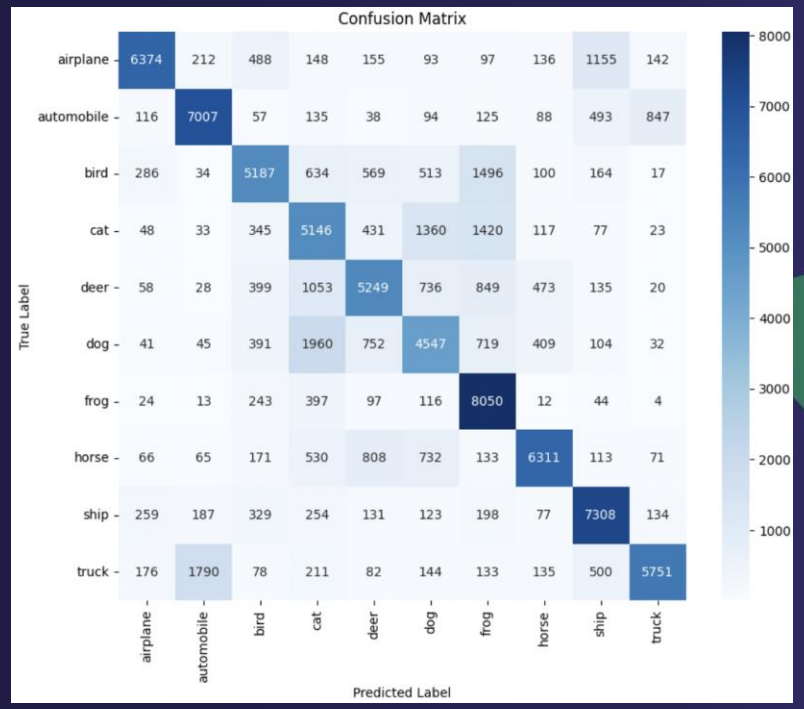
90K TRAIN / 90K VALID / 90K TEST



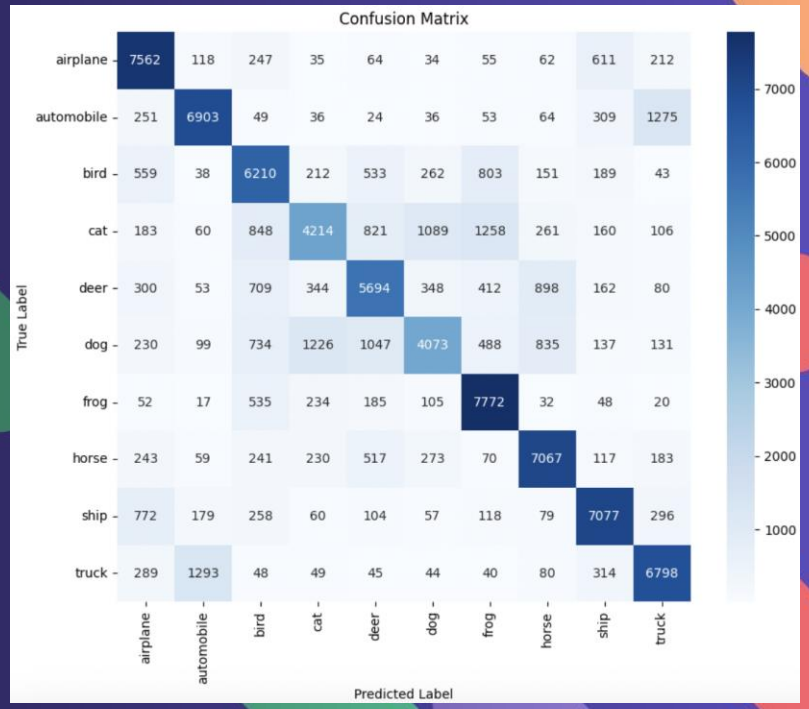
144K TRAIN / 36K VALID / 90K TEST



90K TRAIN / 90K VALID / 90K TEST



144K TRAIN / 36K VALID / 90K TEST



TEST DATA RESULTS

Table 4: Classification Report for Deep CNNs on dataset (90k/90k/90k)

Class	Precision	Recall	F1-Score	Sample size
Airplane	0.86	0.71	0.78	9000
Automobile	0.74	0.78	0.76	9000
Bird	0.67	0.58	0.62	9000
Cat	0.49	0.57	0.53	9000
Deer	0.63	0.58	0.61	9000
Dog	0.54	0.51	0.52	9000
Frog	0.61	0.89	0.72	9000
Horse	0.80	0.70	0.75	9000
Ship	0.72	0.81	0.77	9000
Truck	0.82	0.64	0.72	9000
Accuracy	-	-	0.68	90000

Table 5: Classification Report for Deep CNNs on dataset (144k/36k/90k)

Class	Precision	Recall	F1-Score	Sample size
Airplane	0.72	0.84	**0.78**	9000
Automobile	0.78	0.77	**0.77**	9000
Bird	0.63	0.69	**0.66**	9000
Cat	0.63	0.47	**0.54**	9000
Deer	0.63	0.63	**0.63**	9000
Dog	0.64	0.45	**0.53**	9000
Frog	0.70	0.86	**0.77**	9000
Horse	0.74	0.79	**0.76**	9000
Ship	0.78	0.79	**0.78**	9000
Truck	0.74	0.76	**0.75**	9000
Accuracy	-	-	**0.70**	90000

DEEP CNN WITH REGULARIZATION AND DROPOUT

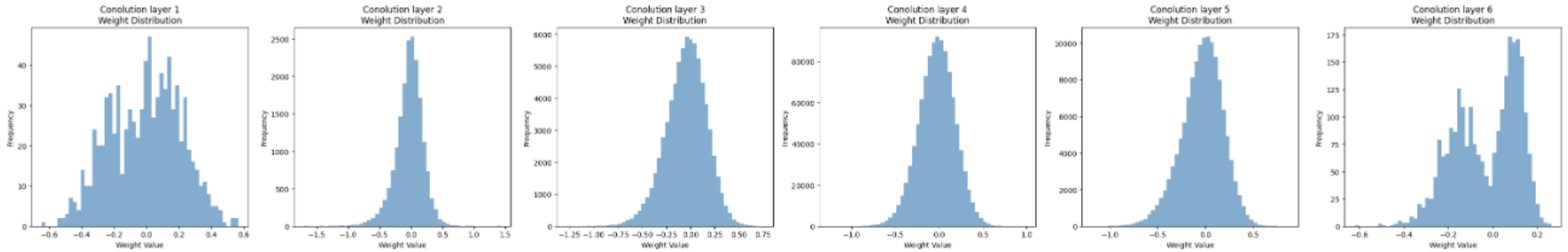


Figure 7: Weight distribution for deep CNN for each convolutional layer

DEEP CNN WITH REGULARIZATION AND DROPOUT AS A WELL TRAINED MODEL

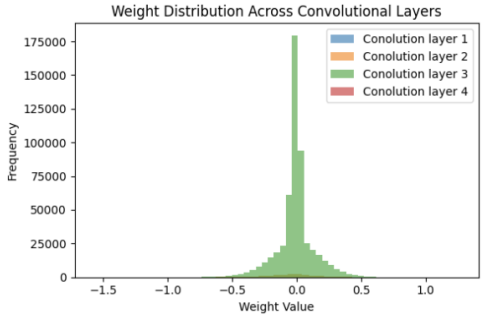


Figure 8: Poorly trained model with double convolution layer

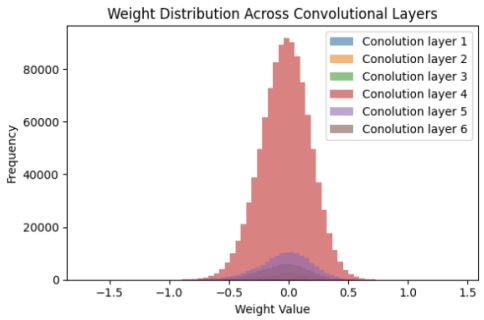
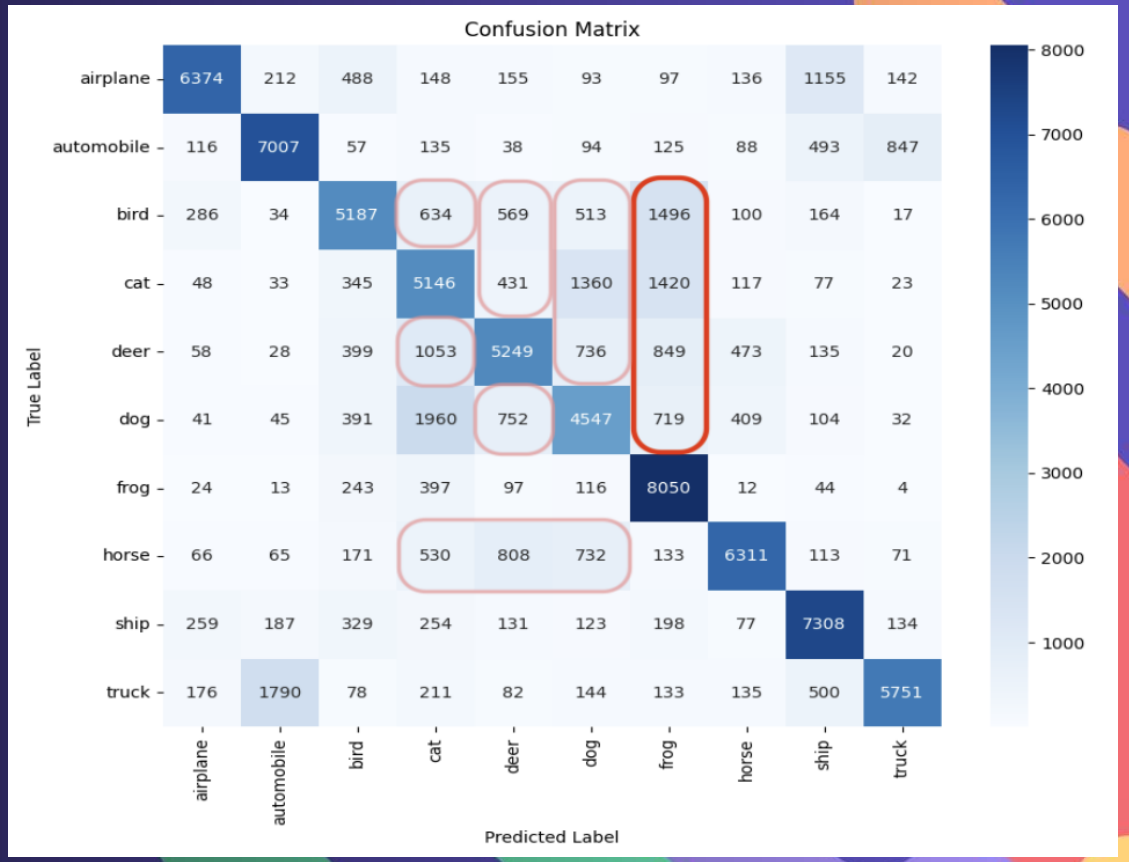
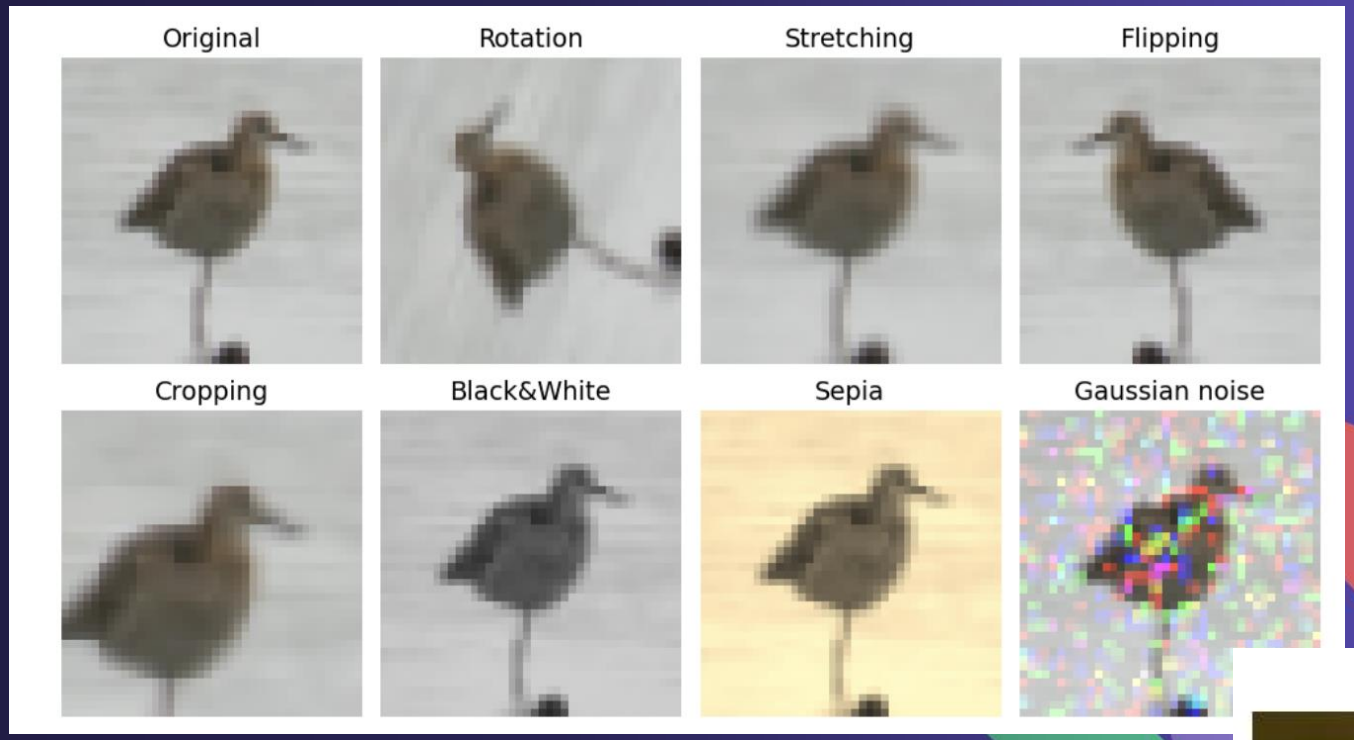


Figure 9: Well trained model with deep CNN architecture

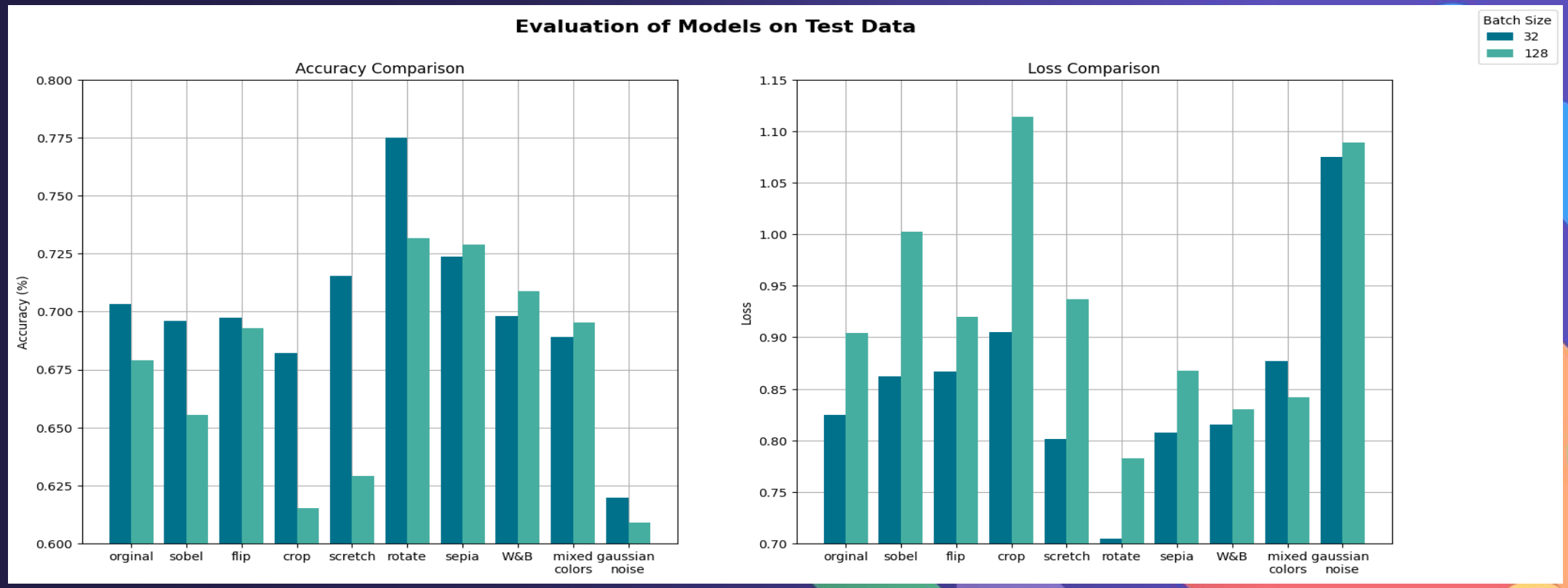
Figure 10: Comparison of weight distribution over all layers

DATA AUGMENTATION

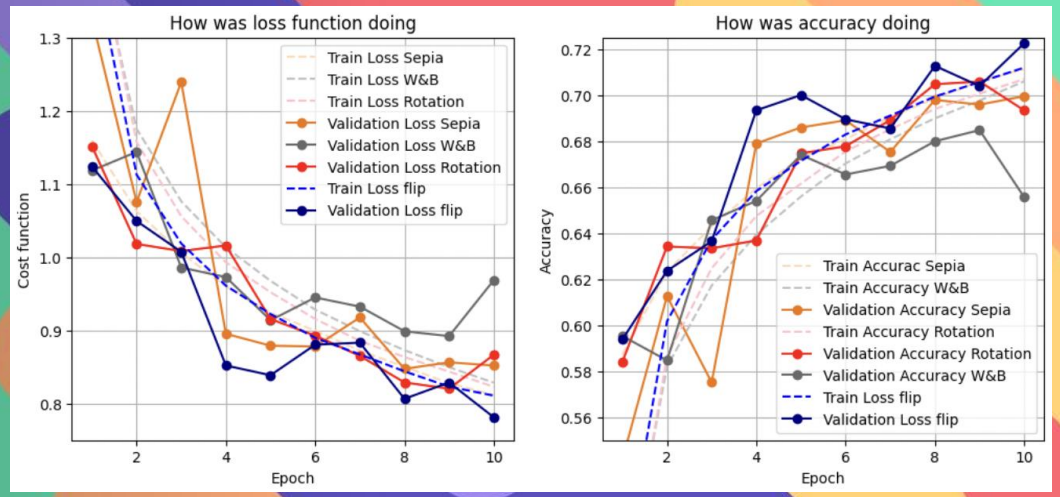




DATA AUGMENTATION



DATA AUGMENTATION

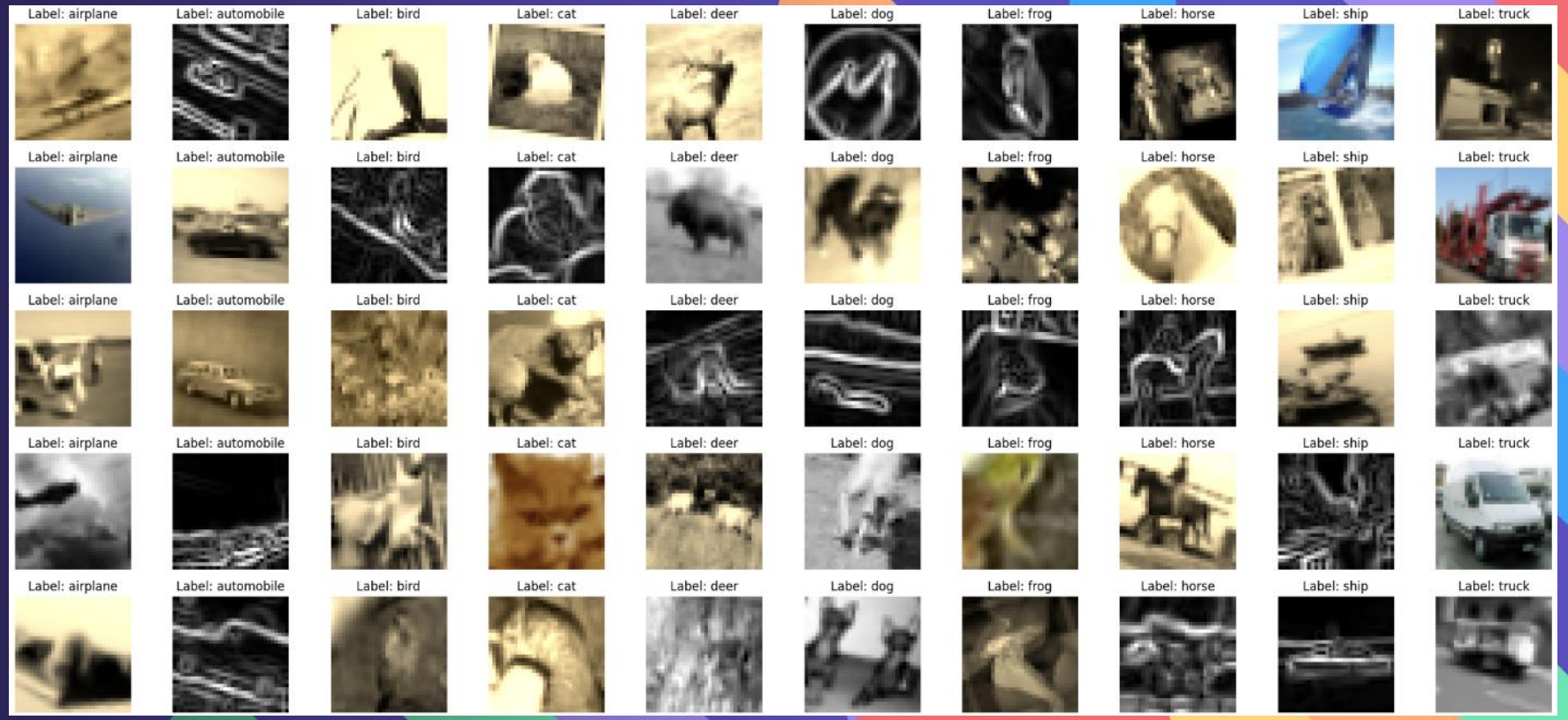


DATA AUGMENTATION – FINAL AUGMENTED DATASET

4 COPIES OF EACH IMAGE:

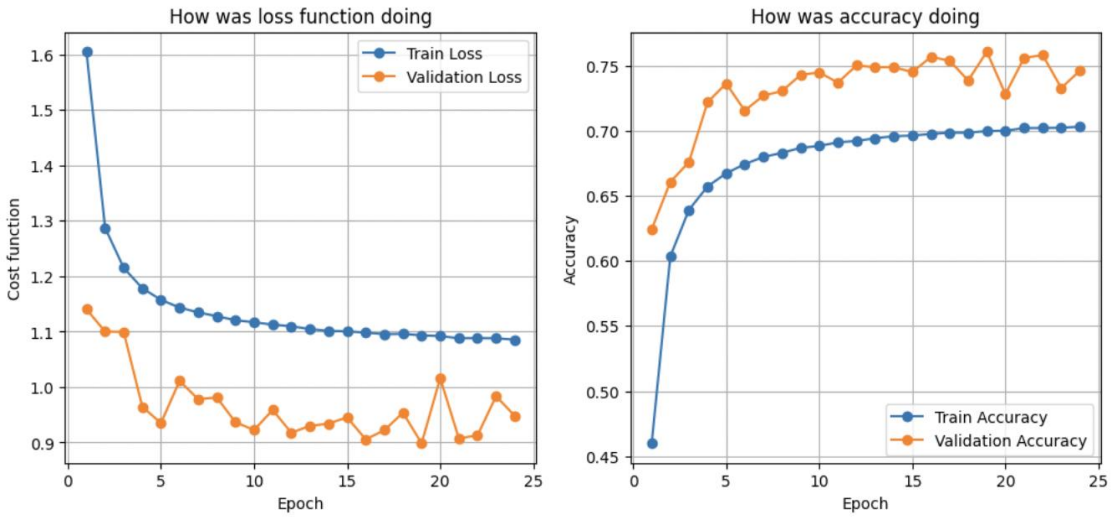
- RANDOM ANGLE 0° – 20°
- FLIP – 50% CHANCE
- SEPIA – 50% CHANCE
- SOBEL – 30% CHANCE
- W&B – 25% CHANCE
- CROP – 20% CHANCE
- STRECH – 20% CHANCE
- GAUSSIAN NOISE – NONE

EVENTUALLY THE TRAINING SET CONTAINED 450K IMAGES

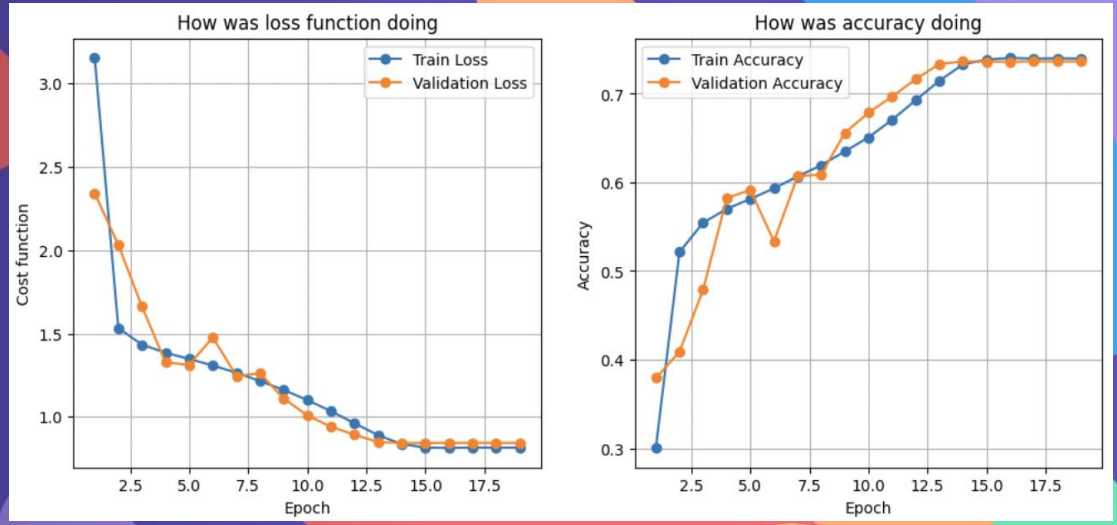


ADVANCED ARCHITECTURES

VISUAL GEOMETRY GROUP (VGG)



RESNET34

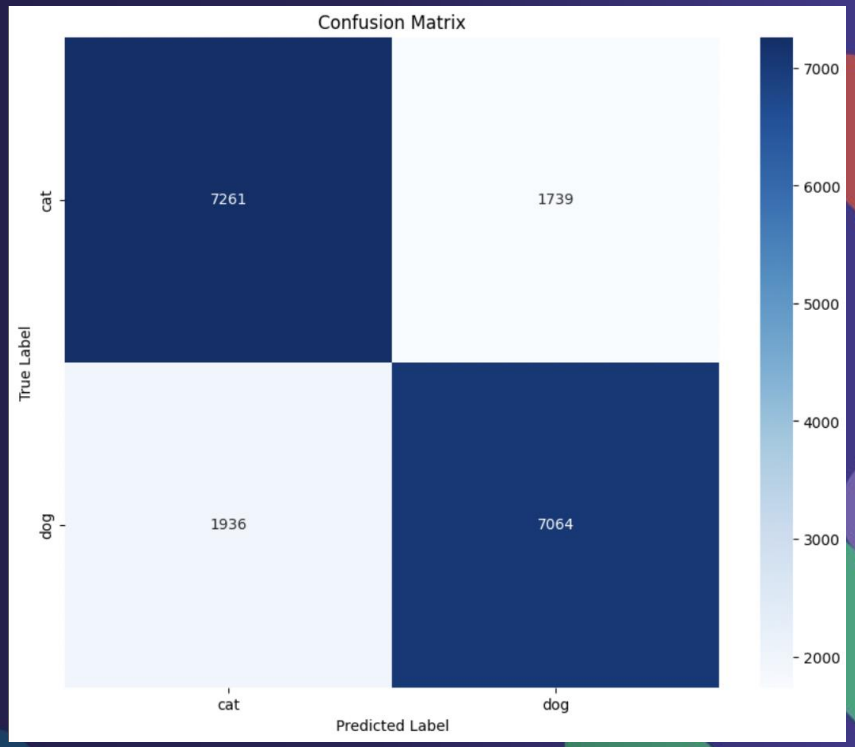


Class	VGG Prec / Rec / F1	ResNet34 Prec / Rec / F1	GoogLeNet Prec / Rec / F1	ResNet18 Prec / Rec / F1	Deep CNN Prec / Rec / F1
airplane	0.85 / 0.83 / 0.84	0.82 / 0.82 / 0.82	0.89 / 0.82 / 0.85	0.87 / 0.85 / **0.86**	0.88 / 0.68 / 0.77
automobile	0.81 / 0.80 / **0.81**	0.79 / 0.78 / 0.78	0.81 / 0.80 / 0.80	0.81 / 0.81 / 0.81	0.79 / 0.77 / 0.78
bird	0.75 / 0.73 / 0.74	0.71 / 0.70 / 0.70	0.73 / 0.75 / 0.74	0.74 / 0.76 / **0.75**	0.60 / 0.70 / 0.65
cat	0.68 / 0.63 / 0.65	0.60 / 0.63 / 0.61	0.74 / 0.55 / 0.63	0.64 / 0.68 / **0.66**	0.50 / 0.65 / 0.56
deer	0.60 / 0.80 / 0.68	0.64 / 0.67 / 0.66	0.68 / 0.73 / 0.70	0.72 / 0.72 / **0.72**	0.63 / 0.59 / 0.61
dog	0.67 / 0.54 / 0.60	0.62 / 0.53 / 0.57	0.62 / 0.64 / **0.63**	0.66 / 0.59 / 0.63	0.60 / 0.41 / 0.49
frog	0.83 / 0.87 / **0.85**	0.80 / 0.83 / 0.81	0.79 / 0.88 / 0.83	0.84 / 0.84 / 0.84	0.70 / 0.88 / 0.78
horse	0.85 / 0.78 / 0.81	0.78 / 0.79 / 0.78	0.76 / 0.83 / 0.80	0.82 / 0.82 / **0.82**	0.79 / 0.73 / 0.76
ship	0.79 / 0.86 / 0.82	0.78 / 0.82 / 0.80	0.81 / 0.85 / 0.83	0.83 / 0.85 / **0.84**	0.71 / 0.83 / 0.77
truck	0.80 / 0.75 / 0.78	0.77 / 0.77 / 0.77	0.81 / 0.76 / 0.78	0.80 / 0.79 / **0.79**	0.83 / 0.67 / 0.74
Accuracy	**0.76**	**0.73**	**0.76**	**0.77**	**0.69**

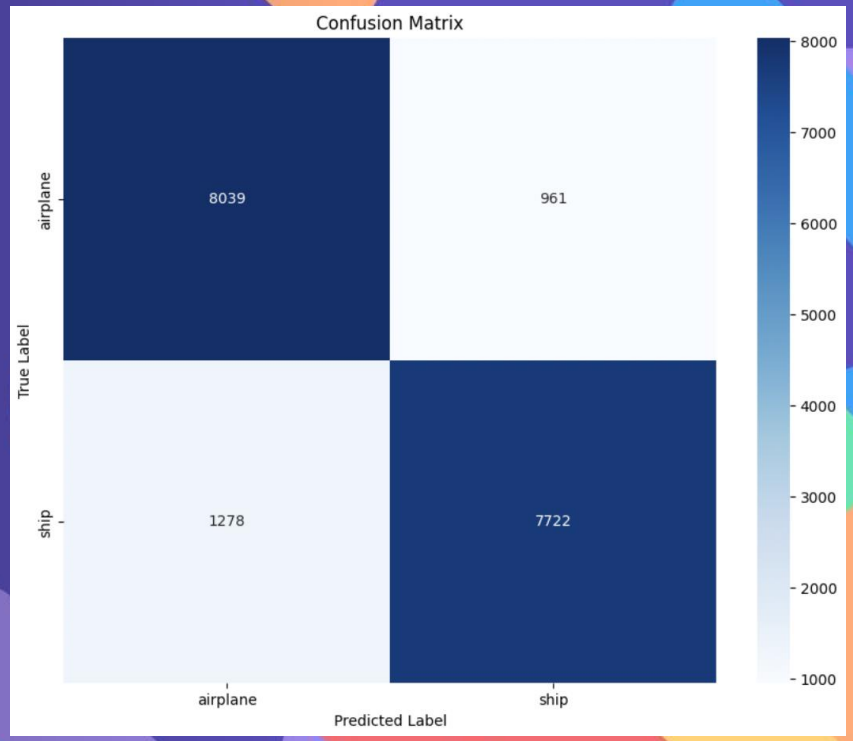
ADVANCED ARCHITECTURES

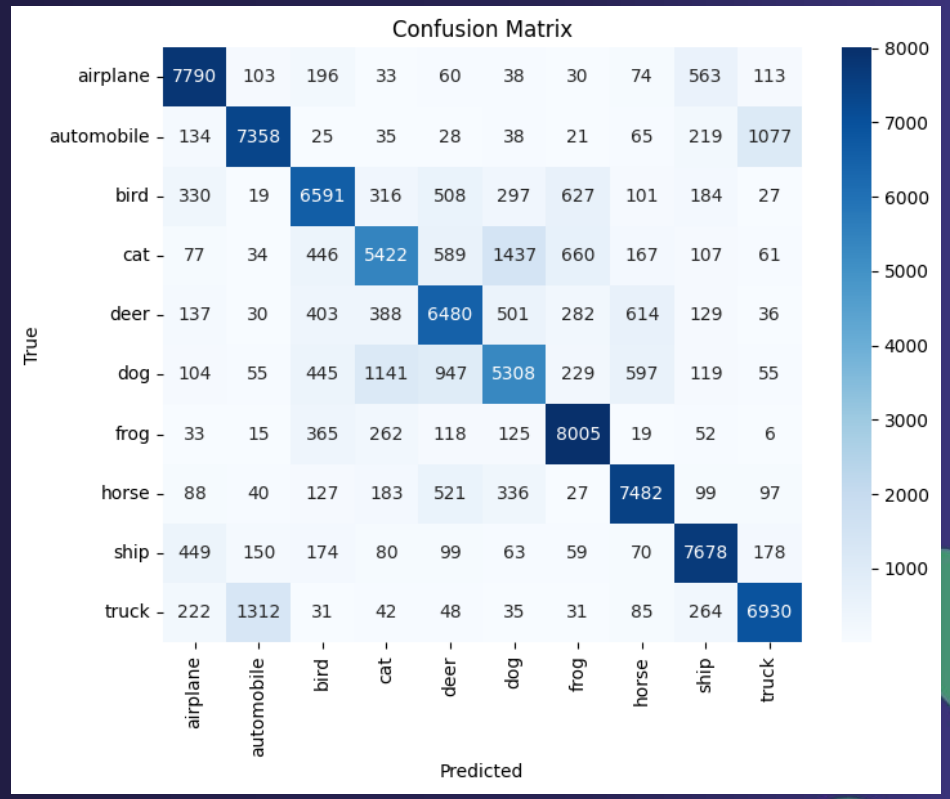
ENSEMBLE

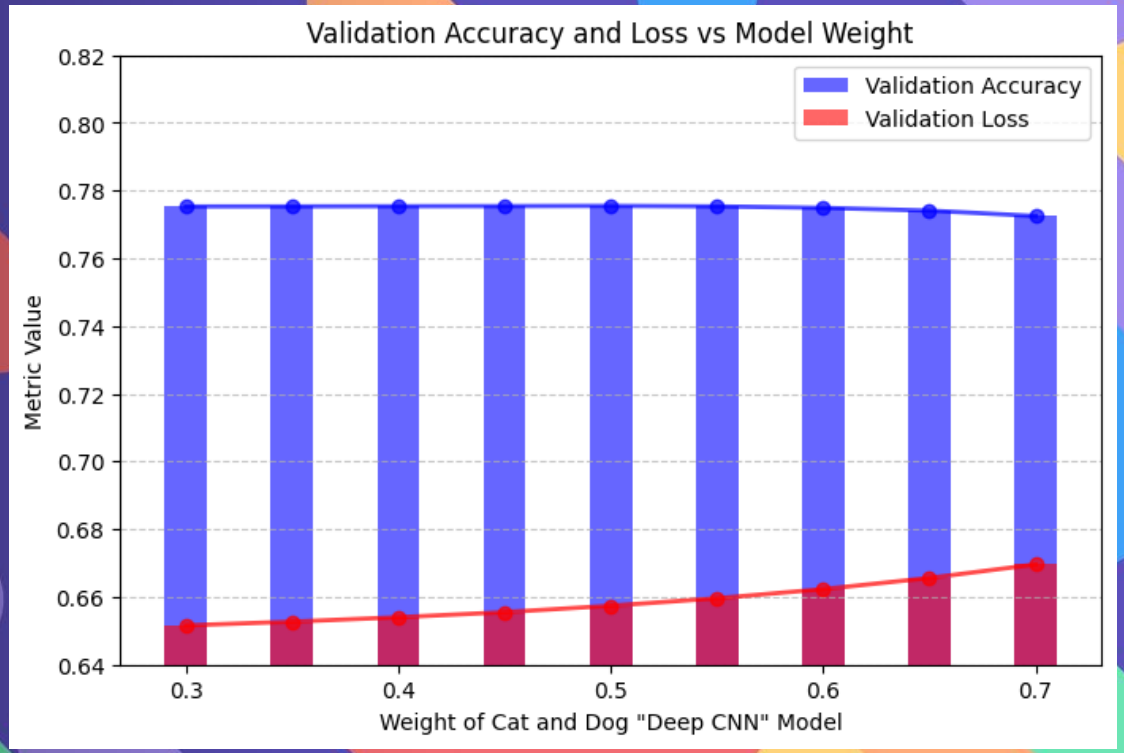
CAT VS DOG



AIRPLANE VS SHIP



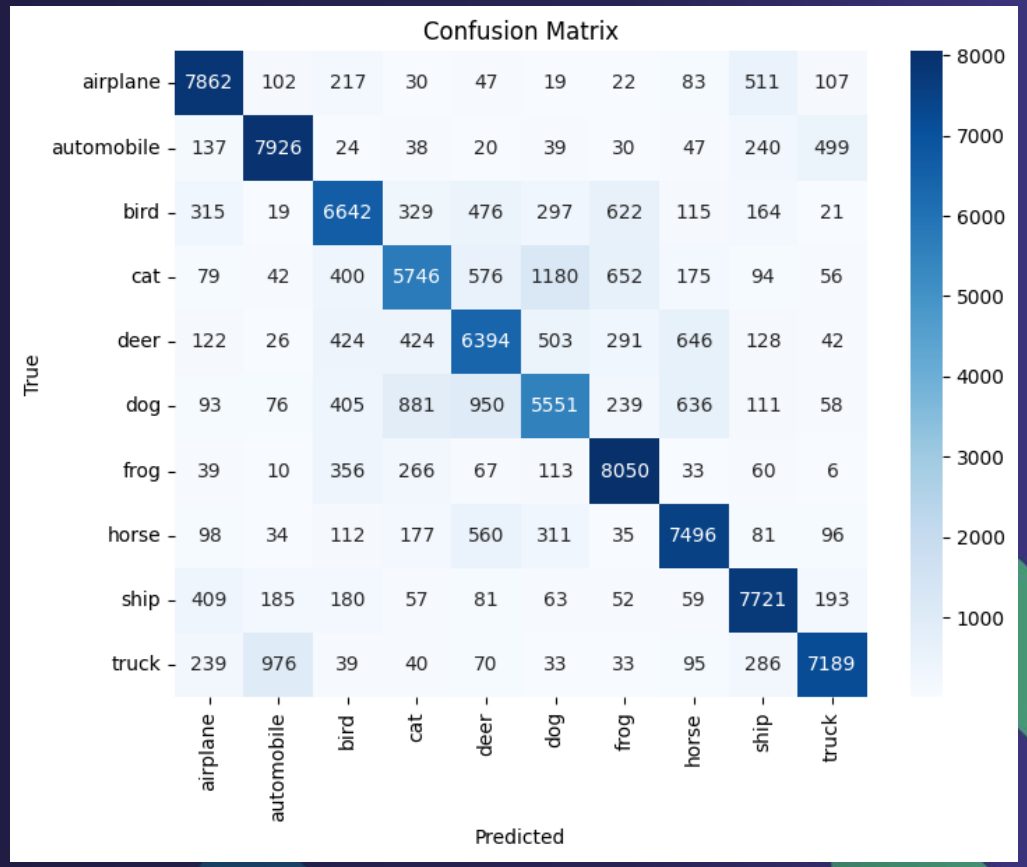




ENSEMBLE

Metric	Value
Accuracy	77.53%
Loss	0.6516

Table 7: Performance metrics of the ensemble model on the test dataset.

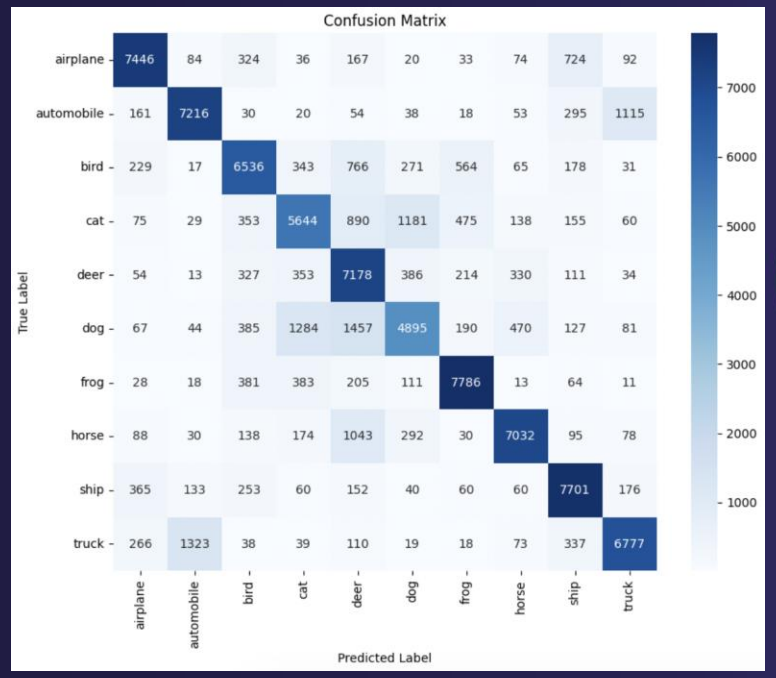


ENSEMBLE

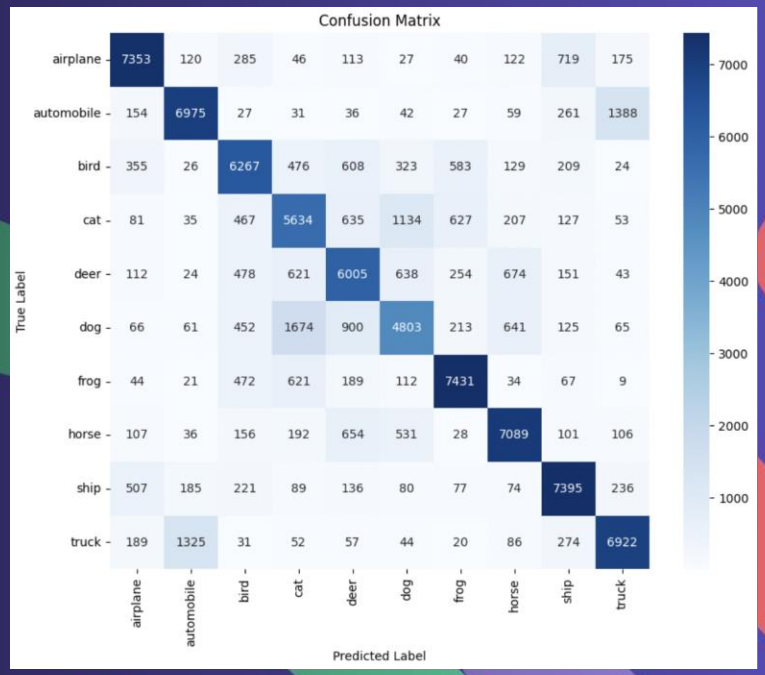
Metric	Value
Accuracy	78.42%
Loss	0.6401

Table 7: Performance metrics of the ensemble model with added "Automobile&truck" model on the test dataset.

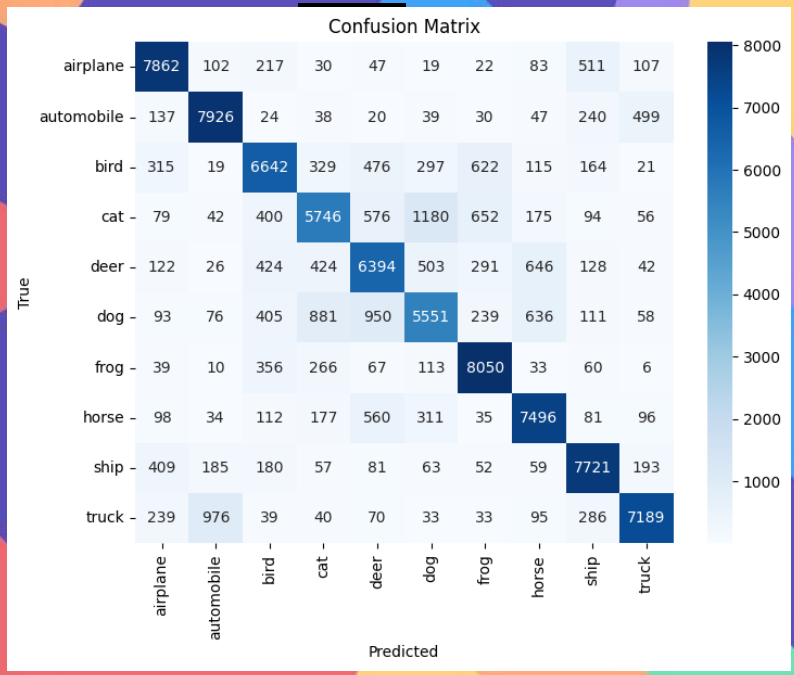
VISUAL GEOMETRY GROUP (VGG) - 75.79% TEST



RESNET18 – 77.23% TEST



ENSEMBLE. - 78.42%



FINAL RESULTS

1) EXPLORE MORE ENSEMBLE MODELS

2) TRY WITH MORE COMPUTING POWER

3) EXPLORE MORE ABOUT RESNET18

FURTHER GOALS

THANK YOU FOR
YOUR ATTENTION