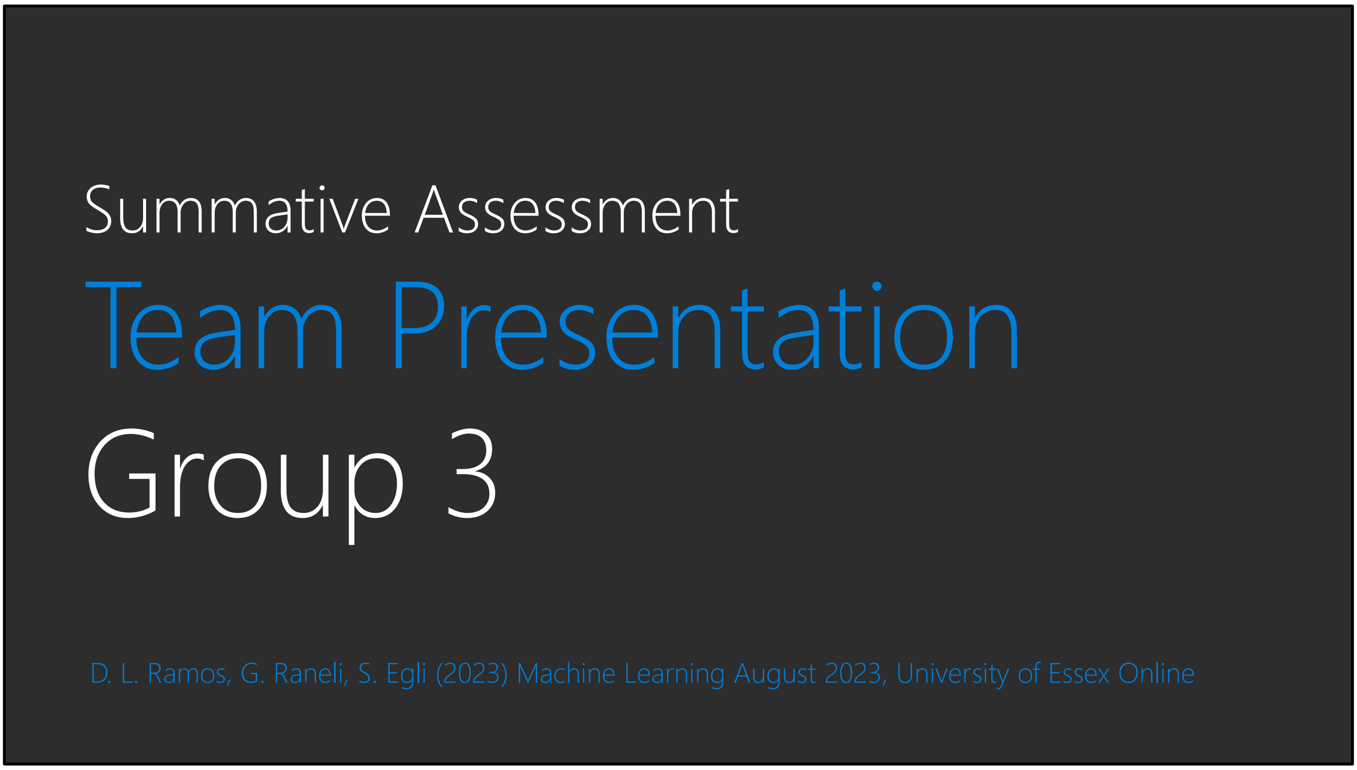


Summative Assessment

Team Presentation

Group 3

D. L. Ramos, G. Raneli, S. Egli (2023) Machine Learning August 2023, University of Essex Online

Welcome to the team presentation by Group 3.

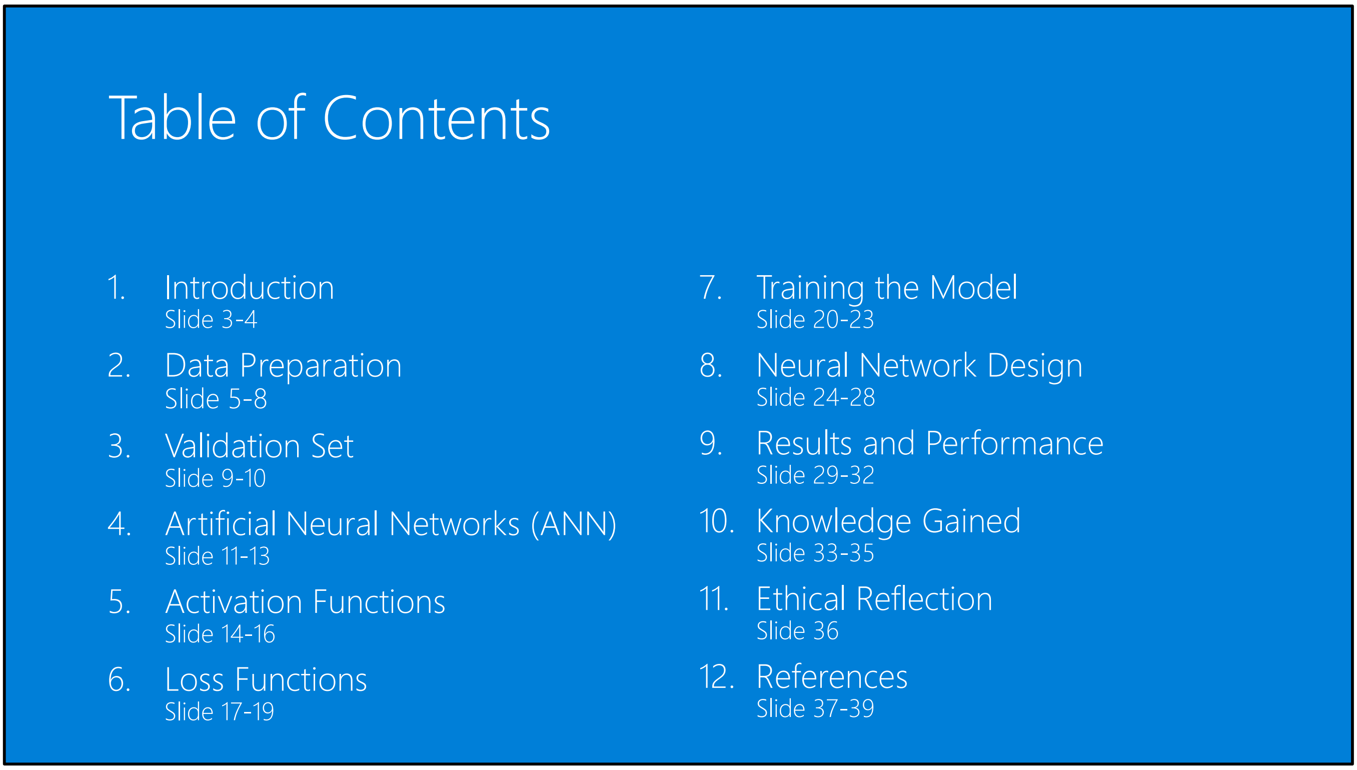
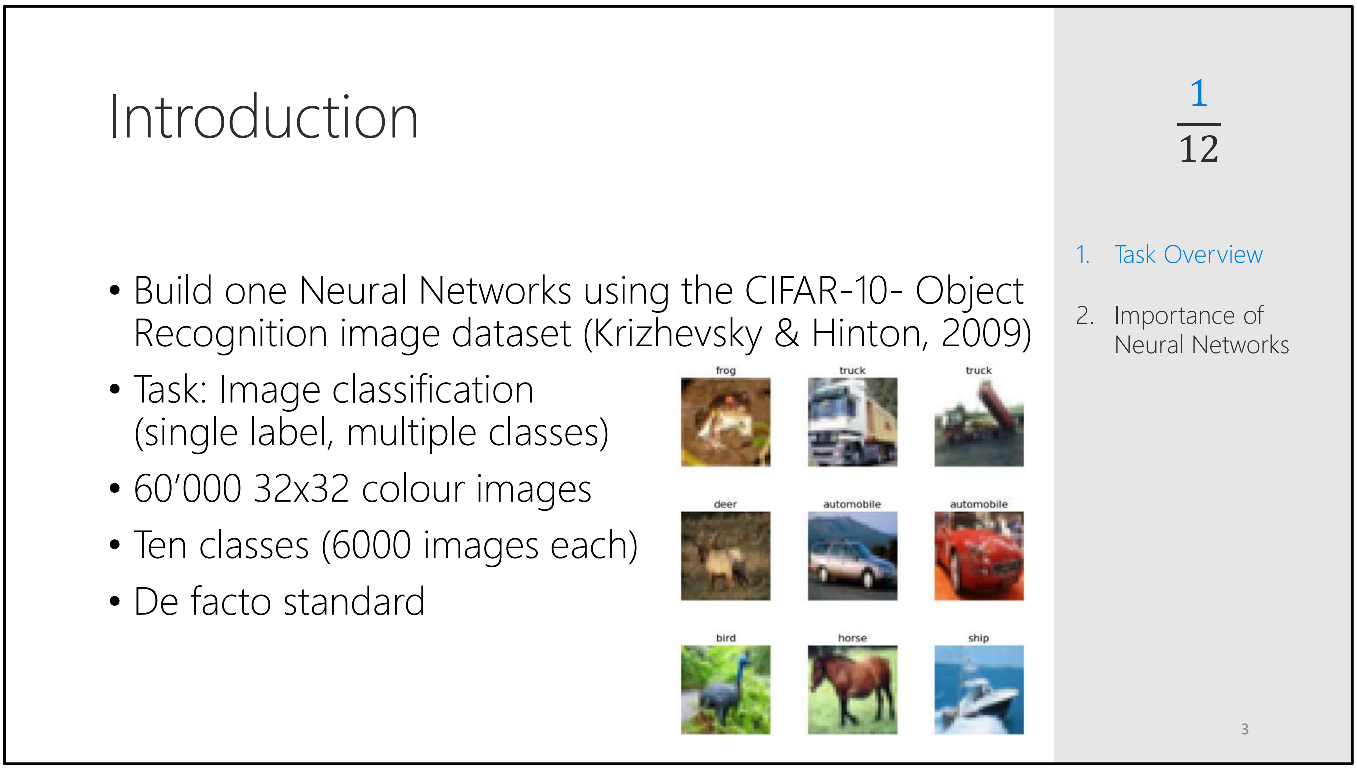


Table of Contents

1. Introduction
   Slide 3-4
2. Data Preparation
   Slide 5-8
3. Validation Set
   Slide 9-10
4. Artificial Neural Networks (ANN)
   Slide 11-13
5. Activation Functions
   Slide 14-16
6. Loss Functions
   Slide 17-19
7. Training the Model
   Slide 20-23
8. Neural Network Design
   Slide 24-28
9. Results and Performance
   Slide 29-32
10. Knowledge Gained
    Slide 33-35
11. Ethical Reflection
    Slide 36
12. References
    Slide 37-39

The agenda comprises 12 points as seen above, with the aim to elucidate team approach, knowledge sharing, and presentation of relevant conclusions.



Introduction

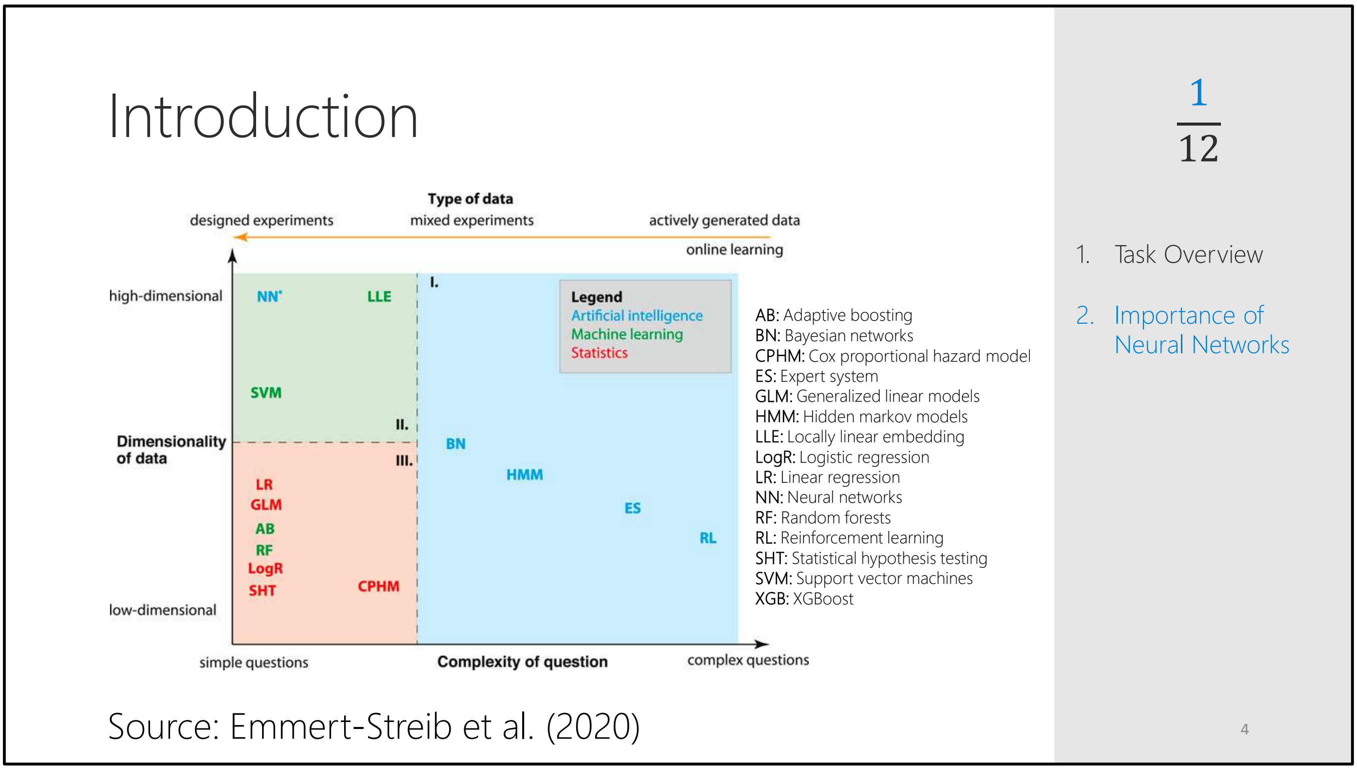
- Build one Neural Networks using the CIFAR-10- Object Recognition image dataset (Krizhevsky & Hinton, 2009)
- Task: Image classification (single label, multiple classes)
- 60’000 32x32 colour images
- Ten classes (6000 images each)
- De facto standard

$\frac{1}{12}$

1. Task Overview
2. Importance of Neural Networks

3

In the constantly evolving world of machine learning, Deep Learning emerges as a process of cutting-edge developments, particularly in image recognition. CIFAR-10, a collection of 60,000 images across ten diverse categories, stands not just as a dataset but as a comprehensive challenge encapsulating the depth and complexity of image classification. This presentation embarks on a profound exploration of creating and refining a deep learning model honed for CIFAR-10, elucidating the deliberate design choices and the rich insights garnered.



Introduction

AB: Adaptive boosting
BN: Bayesian networks
CPHM: Cox proportional hazard model
ES: Expert system
GLM: Generalized linear models
HMM: Hidden markov models
LLE: Locally linear embedding
LogR: Logistic regression
LR: Linear regression
NN: Neural networks
RF: Random forests
RL: Reinforcement learning
SHT: Statistical hypothesis testing
SVM: Support vector machines
XGB: XGBoost

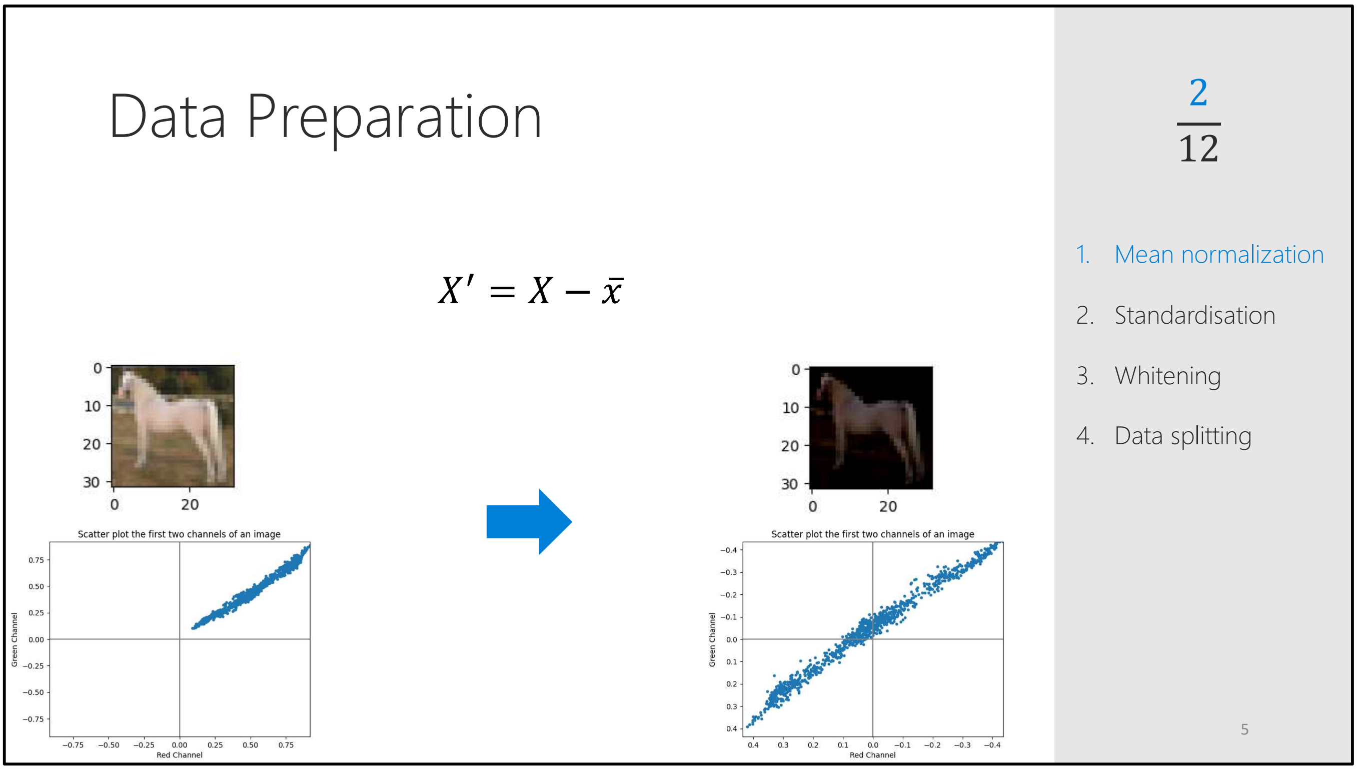
Source: Emmert-Streib et al. (2020)

1
12

1. Task Overview
2. Importance of Neural Networks

4

In data analysis, the disciplines of statistics, machine learning, and artificial intelligence provide multiple methodologies, as Emmert-Streib et al. (2020) argue. Statistical approaches cater to specific challenges, while machine learning is predominantly oriented towards analysing high-dimensional data sets. Concurrently, artificial intelligence endeavours to address multifaceted queries. Before the advent and prominence of deep learning neural networks, machine learning was the predominant methodology within the Region II domain, as shown above. As a notational distinction, Emmert-Streib et al., designated Neural Networks with an asterisk in their taxonomical classification.



Data Preparation

$$X' = X - \bar{x}$$

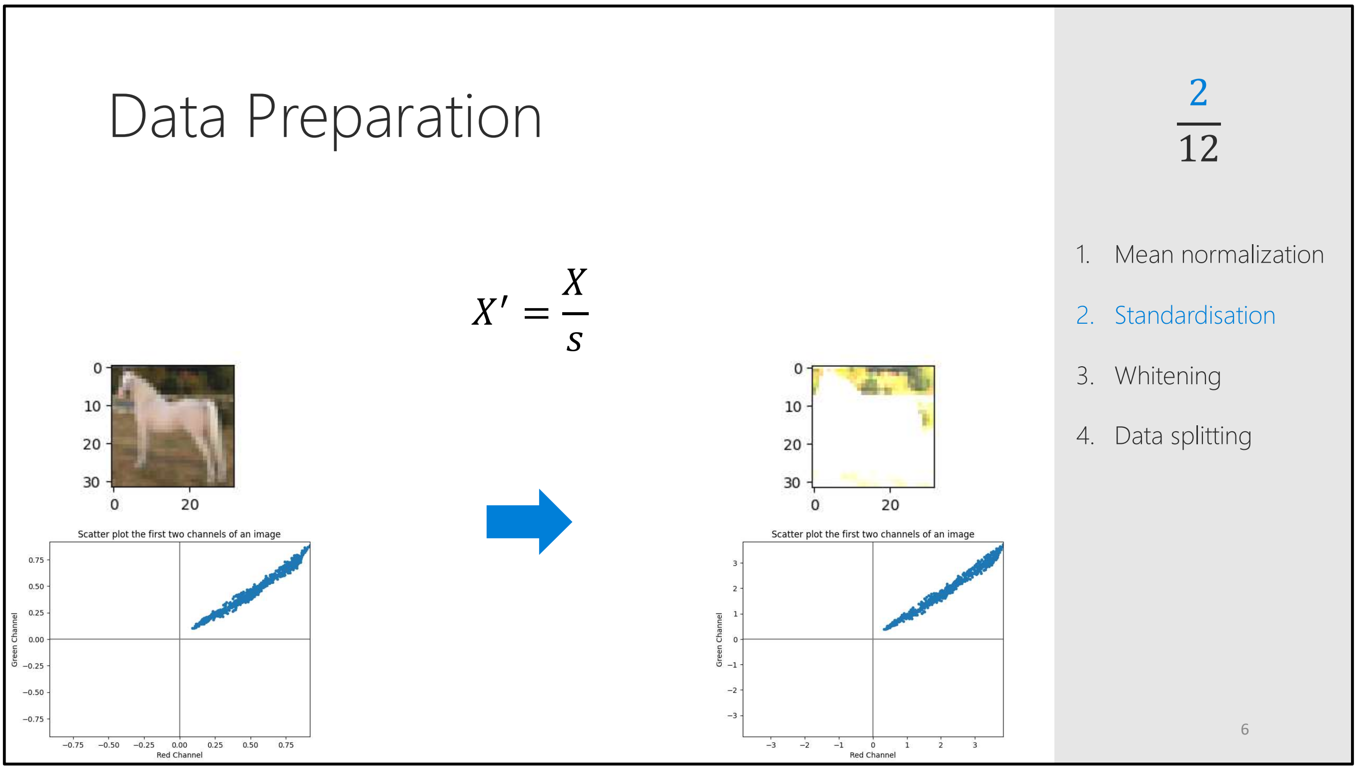
$\frac{2}{12}$

1. Mean normalization
2. Standardisation
3. Whitening
4. Data splitting

5

In the preliminary stages of data preprocessing, each image undergoes a normalisation procedure where it is divided by a constant value of 255, represented as \(X' = \frac{X}{255} \). This process ensures that the resultant values are harmonised to a range optimal for various computational algorithms as argued by Perez and Wang (2017).

Further, mean normalisation is implemented based on the formula provided above. In this formula, \(X' \) represents the normalised dataset, \(X \) refers to the original dataset – which could either be from training, validation, or test sets – and \(\bar{X} \) is the mean vector derived from all features of the training dataset.



Data Preparation

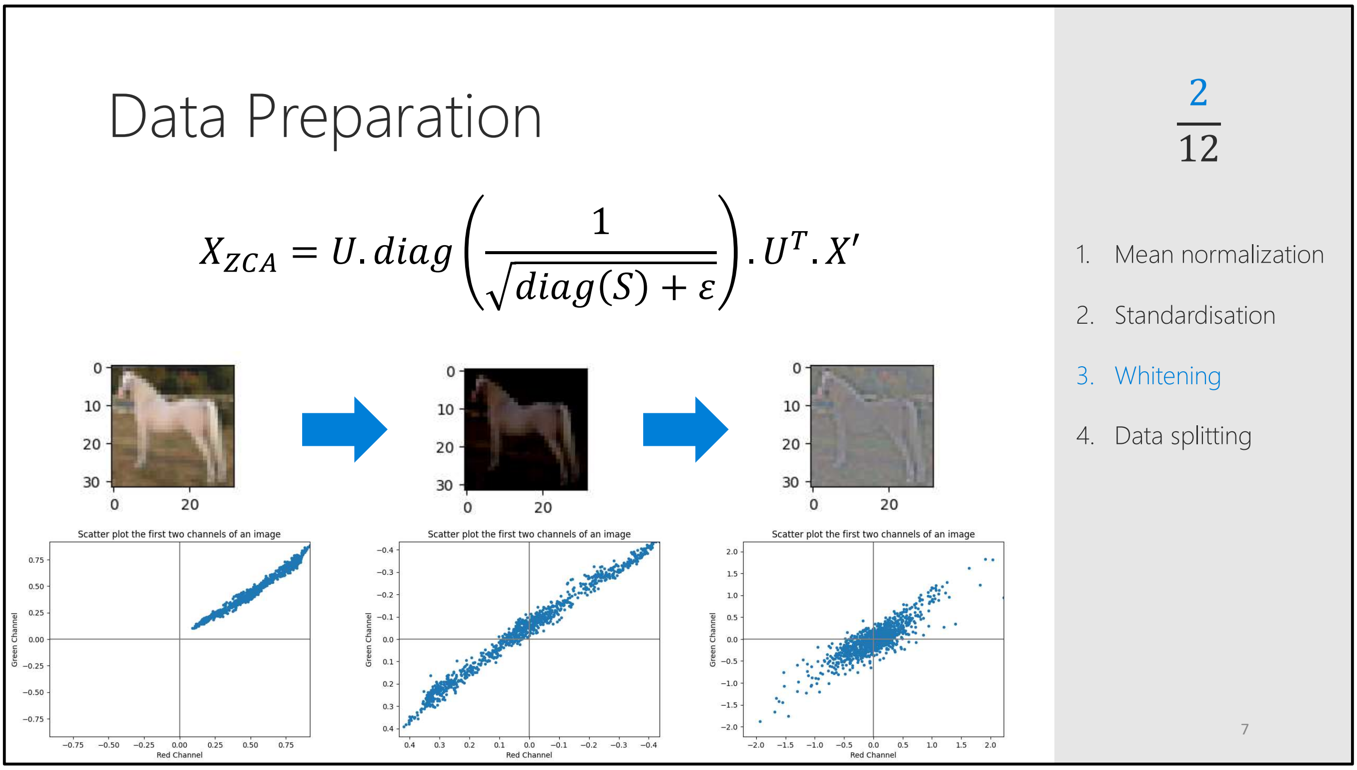
$$X' = \frac{X}{s}$$

$\frac{2}{12}$

1. Mean normalization
2. Standardisation
3. Whitening
4. Data splitting

6

The initial approach contemplated the application of standardisation, wherein the standard deviation of the training set would scale all datasets. Notwithstanding, such a methodology did not produce advantageous outcomes conferming similar results as Huang et al., (2019), leading us to omit its utilisation.



Data Preparation

$$X_{ZCA} = U.diag\left(\frac{1}{\sqrt{diag(S) + \varepsilon}}\right).U^{T}.X'$$

2
12

1. Mean normalization
2. Standardisation
3. Whitening
4. Data splitting

7

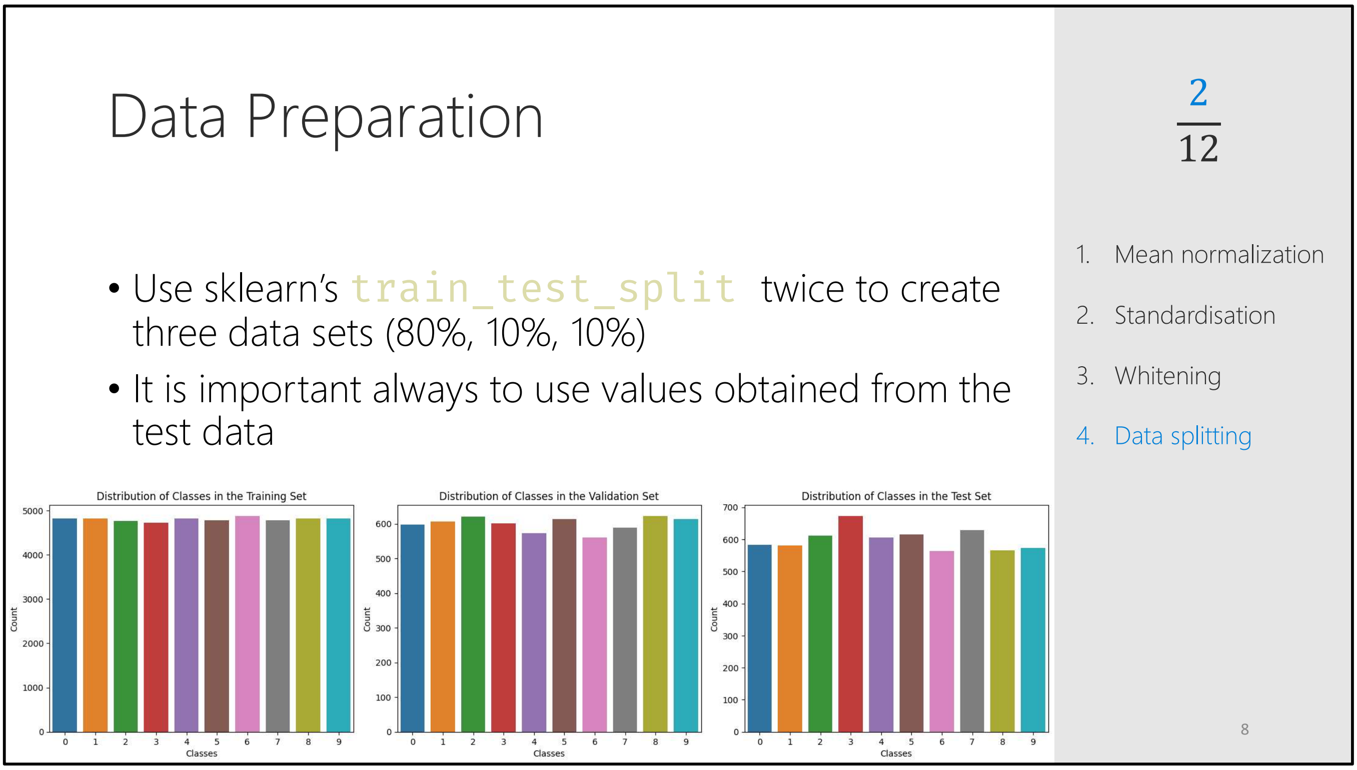
Pal and Sudeep (2016) define whitening as a pre-processing technique to enhance data homogeneity and minimise directional biases. This method transforms data into its eigenbasis and normalises each dimension using its eigenvalue. Geometrically, the whitened data will exhibit a zero mean and an identity covariance matrix for input data adhering to a multivariate Gaussian distribution.

The procedural steps encompass:

1.Mean normalisation of each image.

2.Compute the singular value decomposition about the covariance matrix of the aforementioned mean-normalised data.

3. Whitening application based on the prescribed academic formula, where "diag[...]" denotes the diagonal matrix, "U" is the eigenvector matrix, "S" indicates the eigenvalue matrix, "U^T" is the transposed eigenvector matrix, and "ε" (set at 0.1) is the whitening coefficient.



Data Preparation

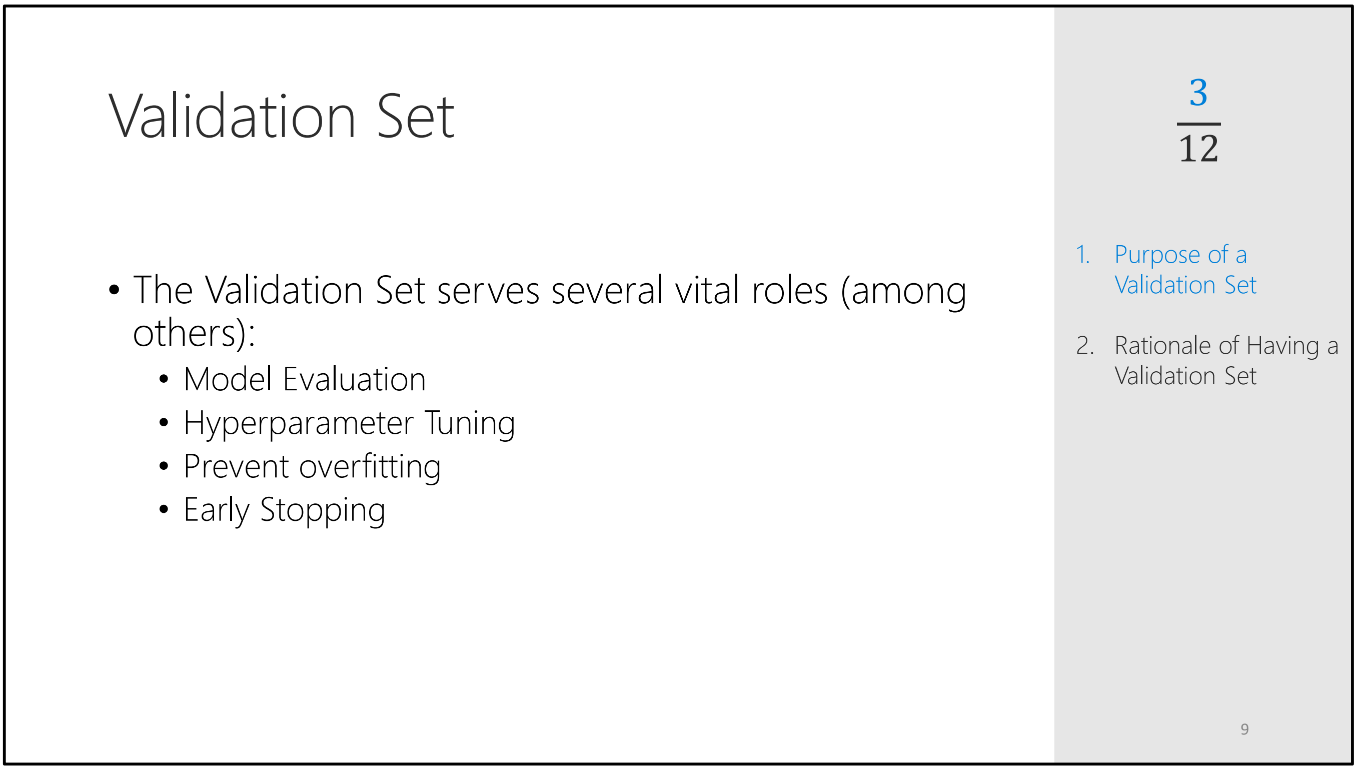
- Use sklearn's `train_test_split` twice to create three data sets (80%, 10%, 10%)
- It is important always to use values obtained from the test data

2
12

1. Mean normalization
2. Standardisation
3. Whitening
4. Data splitting

Utilising Scikit-learn's "train_test_split" function, partitioning the data into three distinct sets is performed: 80% is allocated for training purposes, and the residual 10% is apportioned equally for validation and testing, in adherence to the methodology articulated by Sharma and Phonsa (2021).

It is imperative to emphasise that any pre-processing statistical measures should be exclusively derived from the training dataset and subsequently imposed on the validation and test datasets as argued by Esen et al., (2008). Such a protocol ensures no information leakage transpires from the validation and test sets to the training set. Additionally, all pre-processing protocols undertaken on the training data should be equivalently applied to the validation set to maintain methodological congruence.



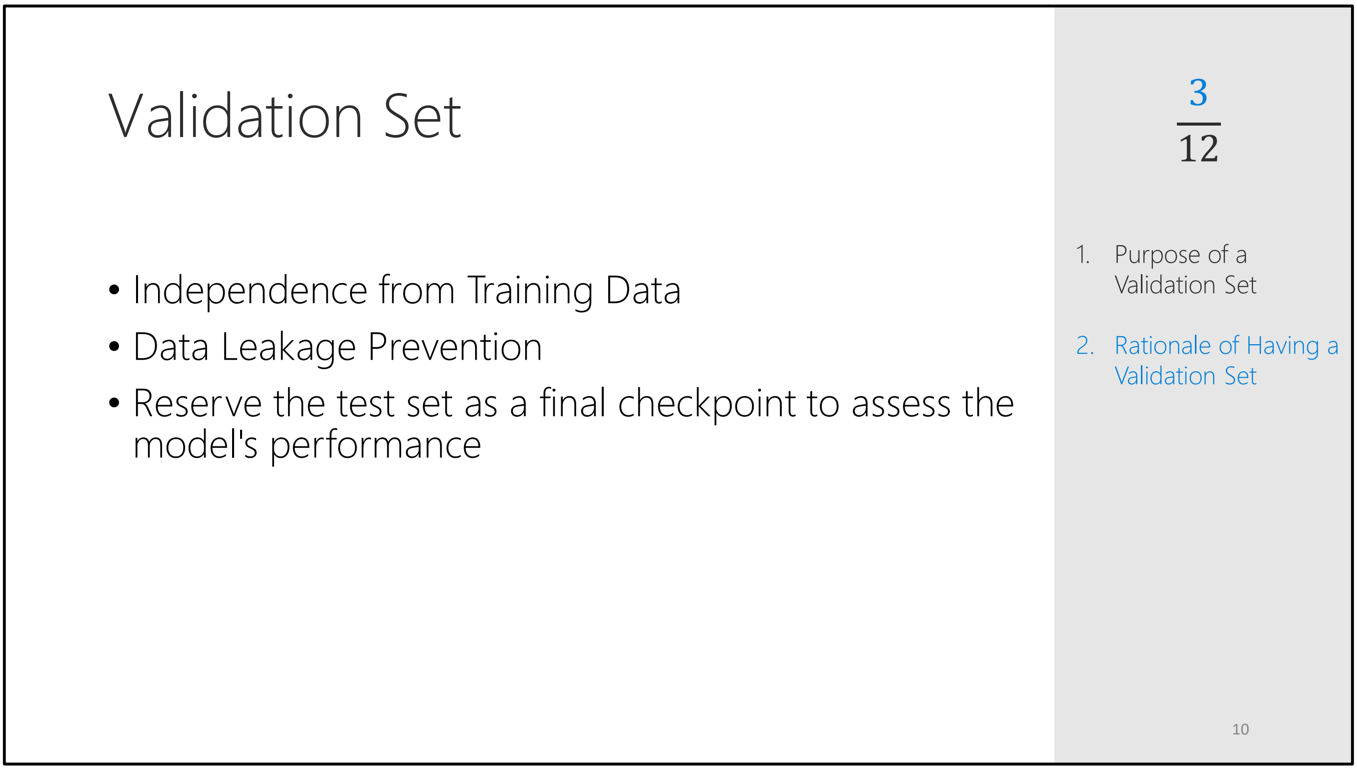
Validation Set

- The Validation Set serves several vital roles (among others):
  - Model Evaluation
  - Hyperparameter Tuning
  - Prevent overfitting
  - Early Stopping

$\frac{3}{12}$

1. Purpose of a Validation Set
2. Rationale of Having a Validation Set

The validation set is pivotal for multiple purposes in machine learning. It aids in assessing how well a model generalises to unseen data, fine-tuning hyperparameters without introducing training bias, detecting and mitigating overfitting, and implementing early stopping when the validation performance plateaus or degrades (Sharma and Phonsa, 2021)



Validation Set

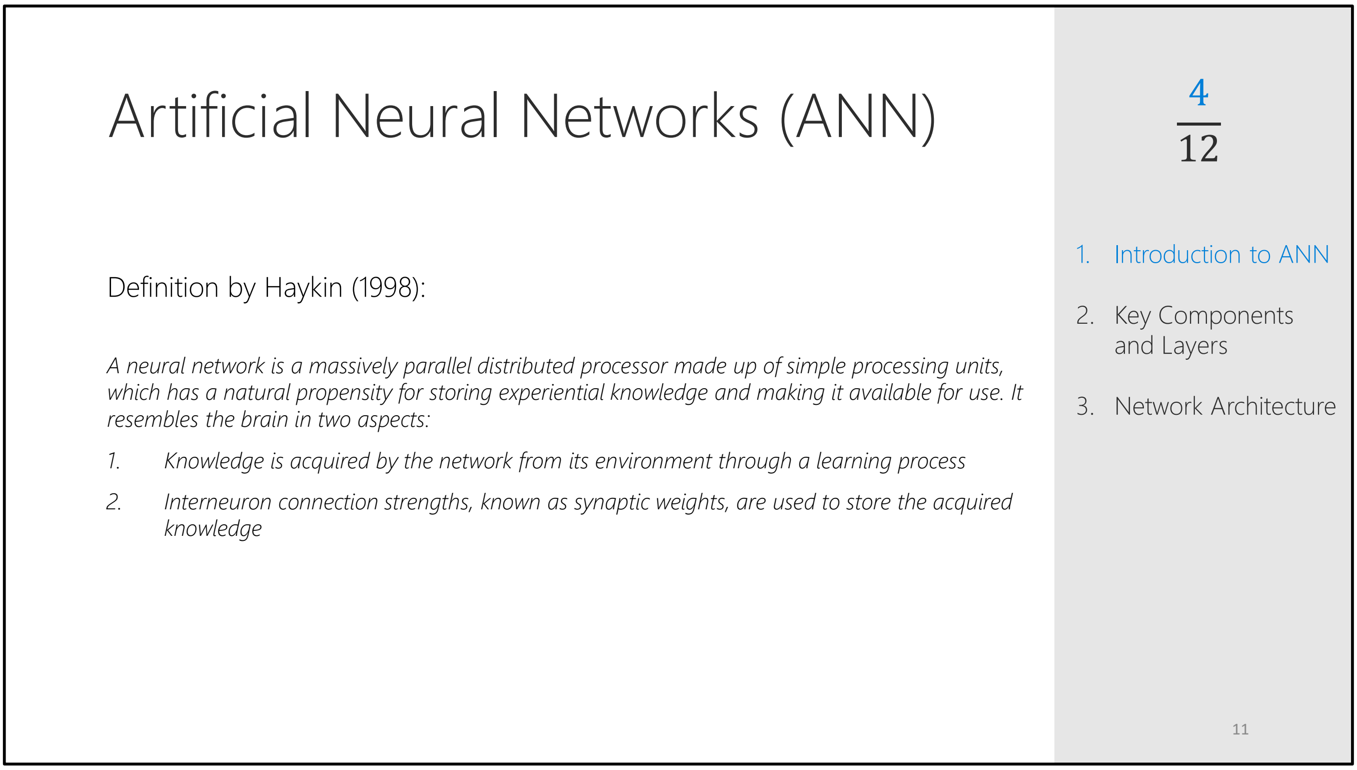
- Independence from Training Data
- Data Leakage Prevention
- Reserve the test set as a final checkpoint to assess the model's performance

$\frac{3}{12}$

1. Purpose of a Validation Set
2. Rationale of Having a Validation Set

10

Leveraging the test dataset as a validation set may cause data leakage. Any determinations informed by the performance metrics of the test dataset, encompassing hyperparameter tuning, run the risk of inadvertently customising the model specifically to that set (Jabbar and Khan, 2015). Such an approach may proffer unduly favourable performance metrics, compromising the model's capacity to extrapolate and adapt to novel, unobserved data, therefore preserving the test set solely to evaluate the model's generalisation performance is academically rigorous.



Artificial Neural Networks (ANN)

$$\frac{4}{12}$$

1. Introduction to ANN
2. Key Components and Layers
3. Network Architecture

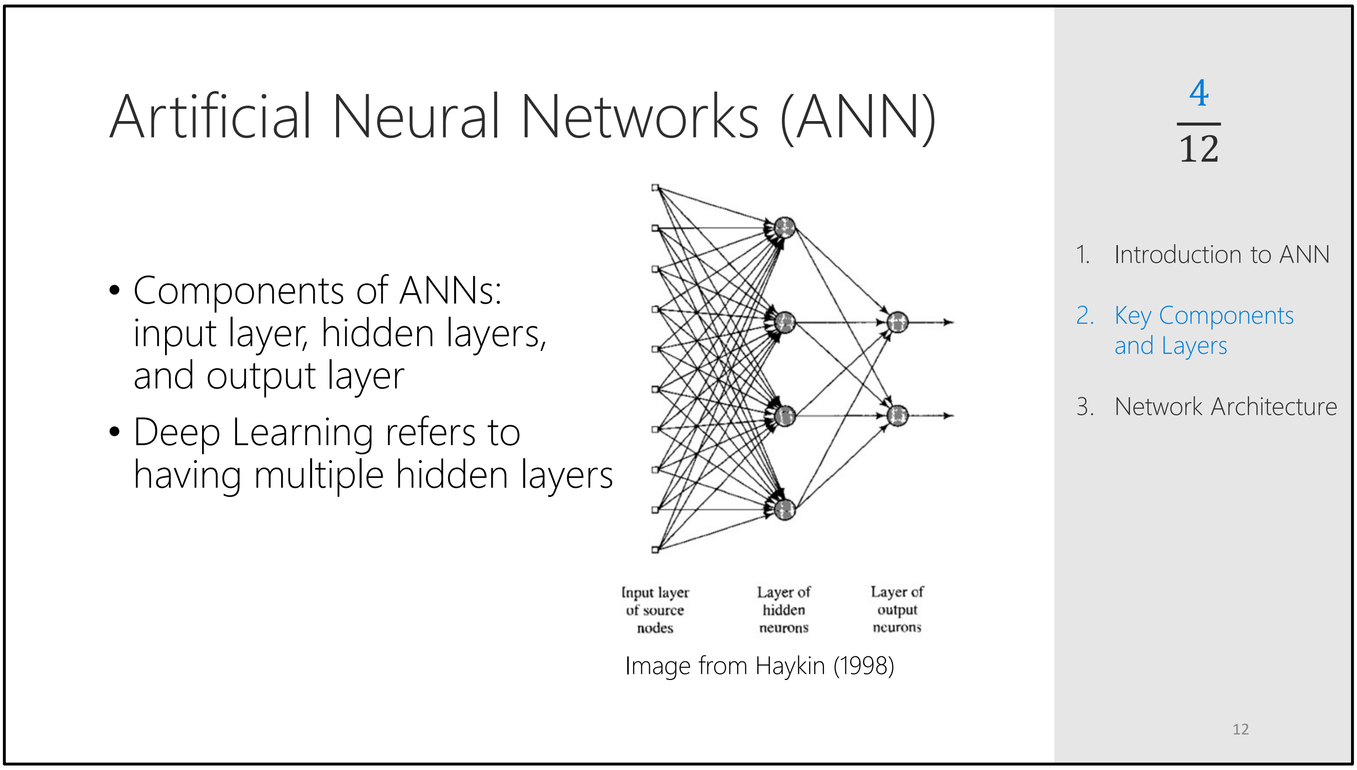
Definition by Haykin (1998):

A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two aspects:

1. *Knowledge is acquired by the network from its environment through a learning process*
2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge*

11

Artificial Neural Networks (ANNs), a part of AI, draw inspiration from the brain but do not precisely mimic it, as Emmert-Streib et al. (2020) noted. ANNs offer advantages, per Hakin (1998), such as nonlinearity, versatility, adaptability to changing environments, confidence in decision-making, and handling contextual information seamlessly



Artificial Neural Networks (ANN)

- Components of ANNs: input layer, hidden layers, and output layer
- Deep Learning refers to having multiple hidden layers

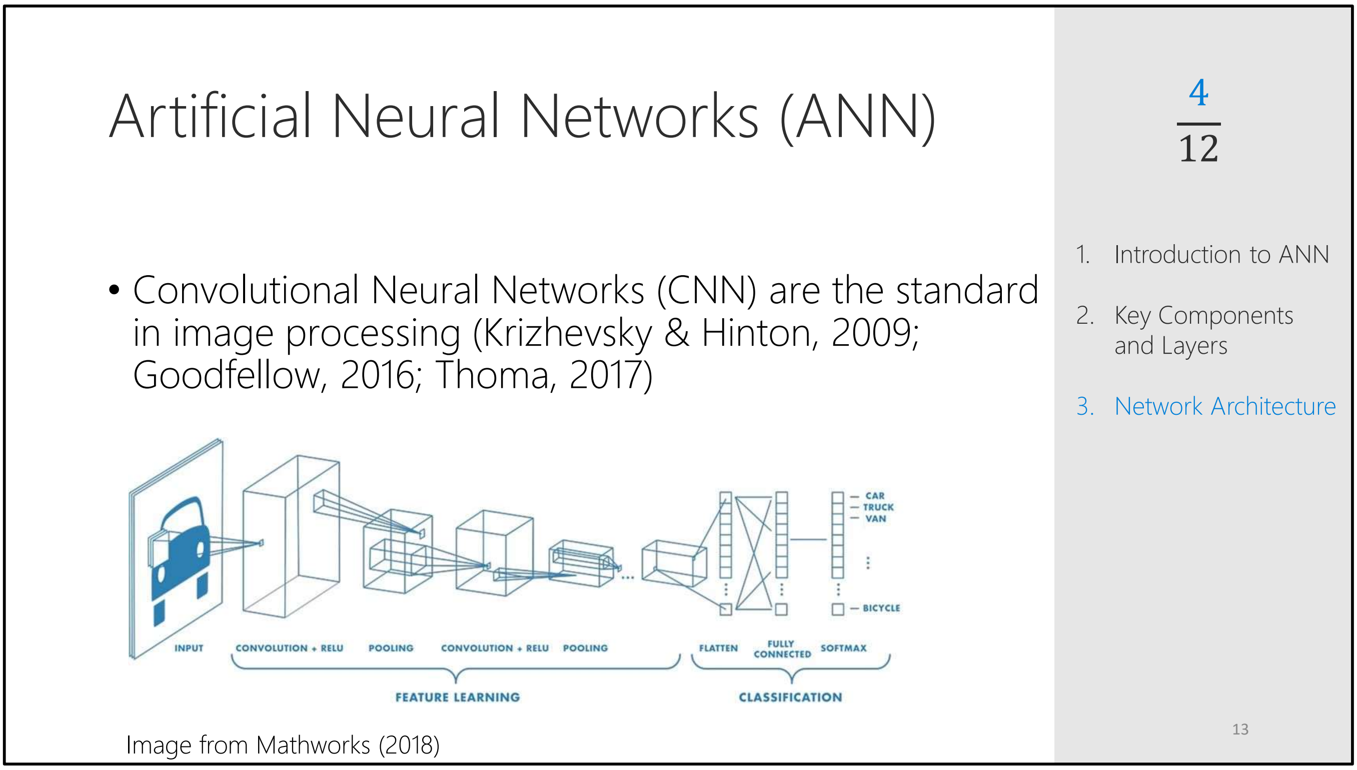
Image from Haykin (1998)

$\frac{4}{12}$

1. Introduction to ANN
2. Key Components and Layers
3. Network Architecture

12

ANNs typically include an input, multiple hidden, and output layers. Multiple hidden layers result in "deep learning." ANNs serve various purposes, from regression to natural language processing and dimensionality reduction (e.g., Autoencoders). Despite their differences, all ANNs share standard components: layers, activation functions, and weights/parameters.



Artificial Neural Networks (ANN)

$\frac{4}{12}$

1. Introduction to ANN
2. Key Components and Layers
3. Network Architecture

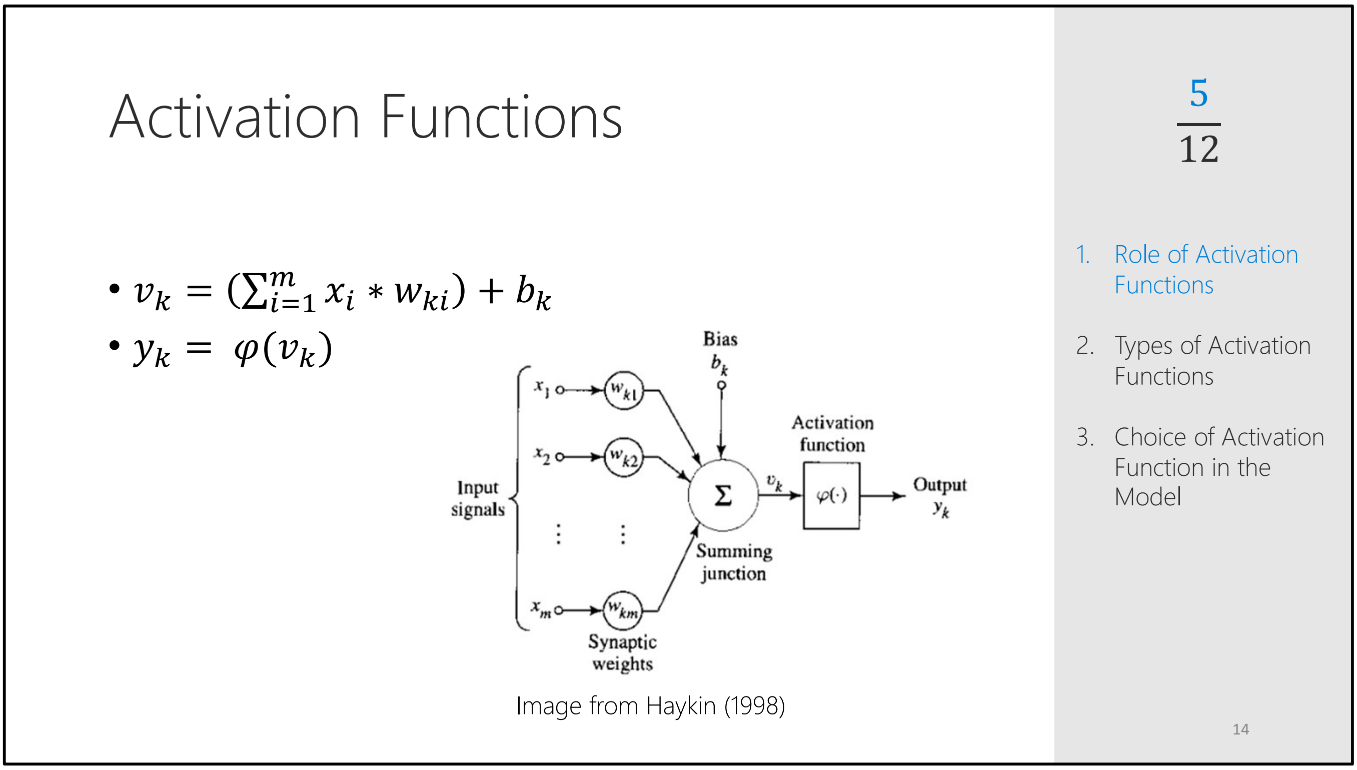
- Convolutional Neural Networks (CNN) are the standard in image processing (Krizhevsky & Hinton, 2009; Goodfellow, 2016; Thoma, 2017)

Image from Mathworks (2018)

13

Several authors (Krizhevsky & Hinton (2009); Goodfellow (2016); Thoma (2017)) recommend a network architecture that combines convolutional and pooling layers for feature learning, coupled with classic feedforward layers for classification.

Convolutional layers are tailored to capture local patterns and features within input images, effectively detecting edges, textures, and intricate visual elements that aid in comprehending the image's structure. Complementary to convolutional layers, pooling layers play a pivotal role in reducing the spatial dimensions (width and height) of feature maps generated by the convolutional layers.



Activation Functions

- $v_k = (\sum_{i=1}^{m} x_i * w_{ki}) + b_k$
- $y_k = \varphi(v_k)$

Image from Haykin (1998)

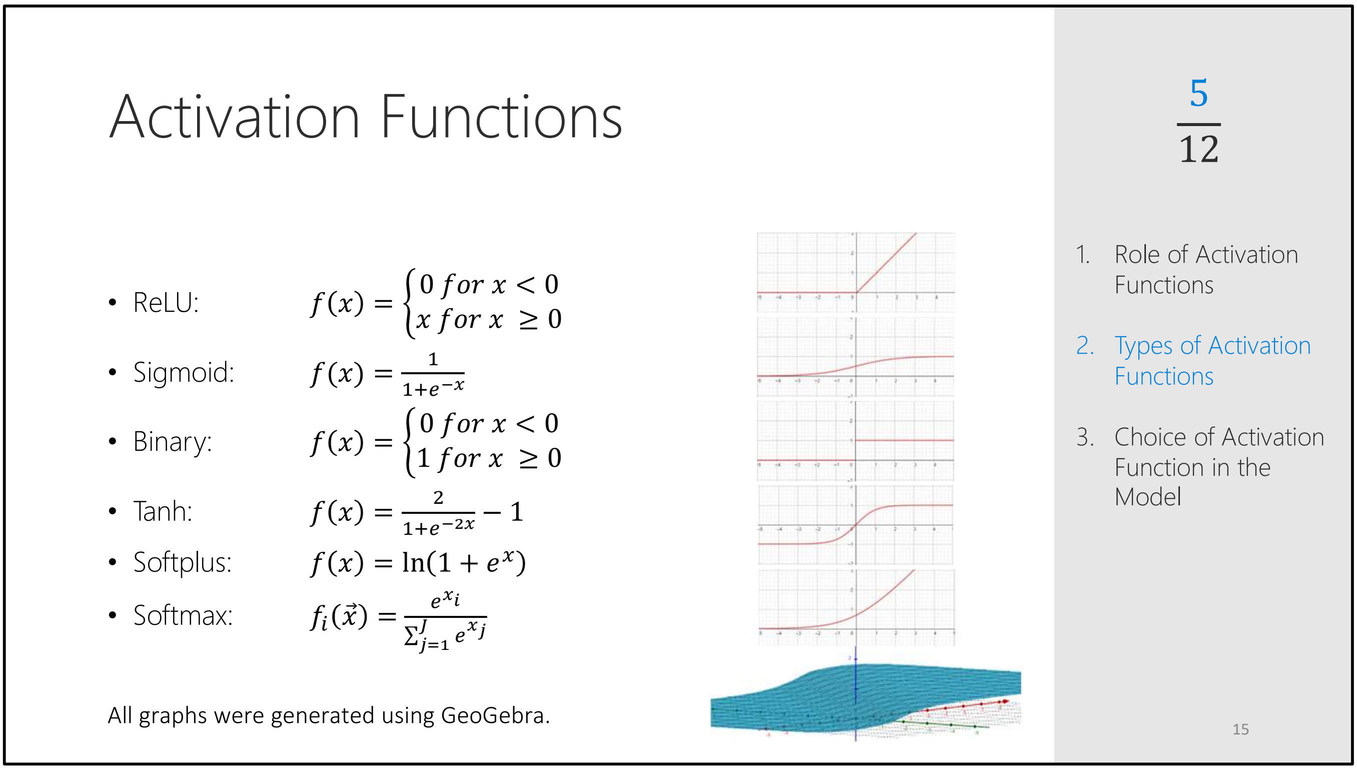
$\frac{5}{12}$

1. Role of Activation Functions
2. Types of Activation Functions
3. Choice of Activation Function in the Model

14

During training, the neural network updates the weights and biases of its neurons. If the generated output significantly differs from the actual value, the network computes the error and adjusts the weights and biases in a "back-propagation."

In this context, v_k represents the input to the activation function and is essentially a linear transformation of input signals (x_1 to x_m) using weights and bias. Without an activation function, a neural network functions as a linear regression model, limiting its capacity to learn complex patterns from data. Thus, the introduction of non-linearity in the network is achieved through the activation function.



Activation Functions

$\frac{5}{12}$

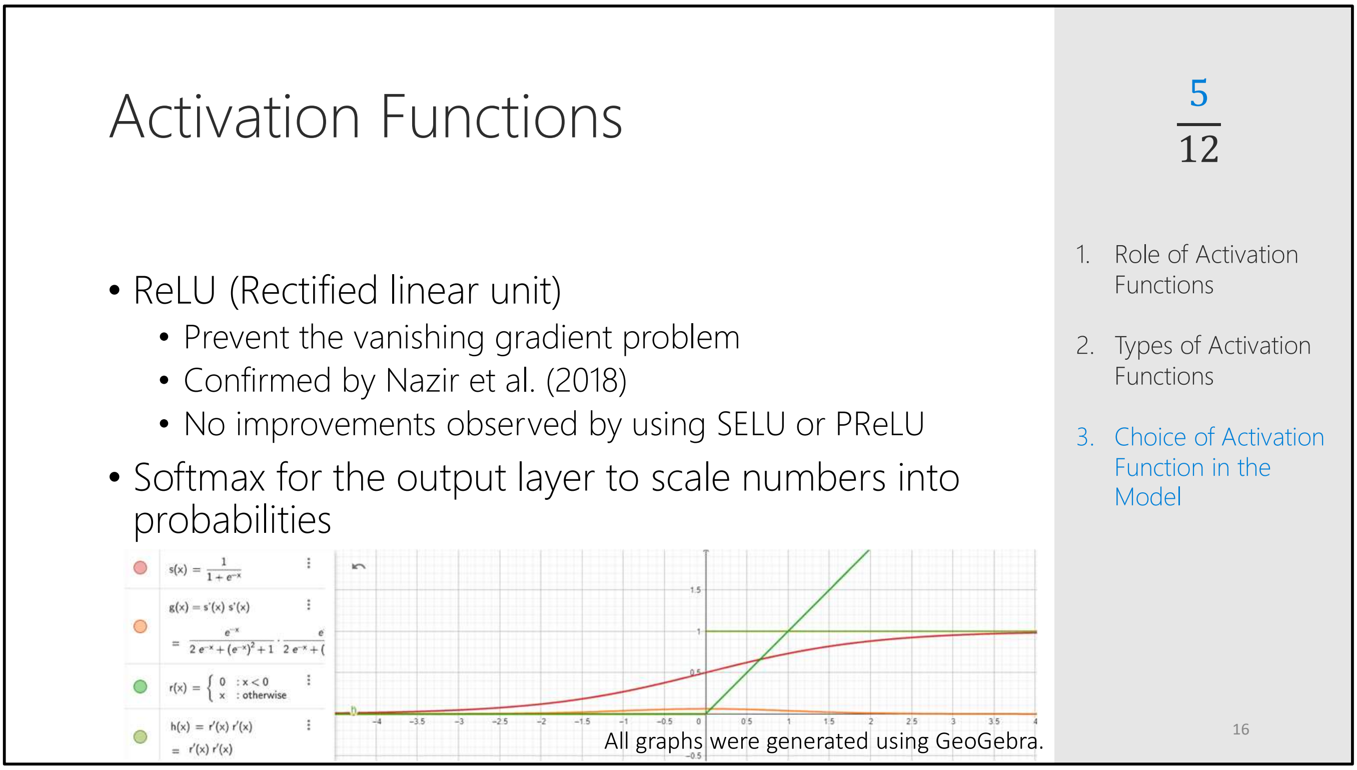
1. Role of Activation Functions
2. Types of Activation Functions
3. Choice of Activation Function in the Model

- ReLU: $f(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$
- Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- Binary: $f(x) = \begin{cases} 0 \; for \; x < 0 \\ 1 \; for \; x \geq 0 \end{cases}$
- Tanh: $f(x) = \frac{2}{1+e^{-2x}} - 1$
- Softplus: $f(x) = \ln(1 + e^{x})$
- Softmax: $f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$

All graphs were generated using GeoGebra.

15

There are numerous types of activation functions. Several of them are depicted here, generated using GeoGebra (2023). Feel free to pause the video to explore further.



Activation Functions

- ReLU (Rectified linear unit)
  - Prevent the vanishing gradient problem
  - Confirmed by Nazir et al. (2018)
  - No improvements observed by using SELU or PReLU
- Softmax for the output layer to scale numbers into probabilities

$s(x) = \frac{1}{1+e^{-x}}$
$g(x) = s'(x)\, s'(x)$
$= \frac{e^{-x}}{2e^{-x}+(e^{-x})^2+1} \cdot \frac{e}{2e^{-x}+(}$
$r(x) = \begin{cases} 0 & : x < 0 \\ x & : \text{otherwise} \end{cases}$
$h(x) = r'(x)\, r'(x)$
$= r'(x)\, r'(x)$
1.5
1
0.5
−0.5
−4
−3.5
−3
−2.5
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5
3
3.5
4

All graphs were generated using GeoGebra.

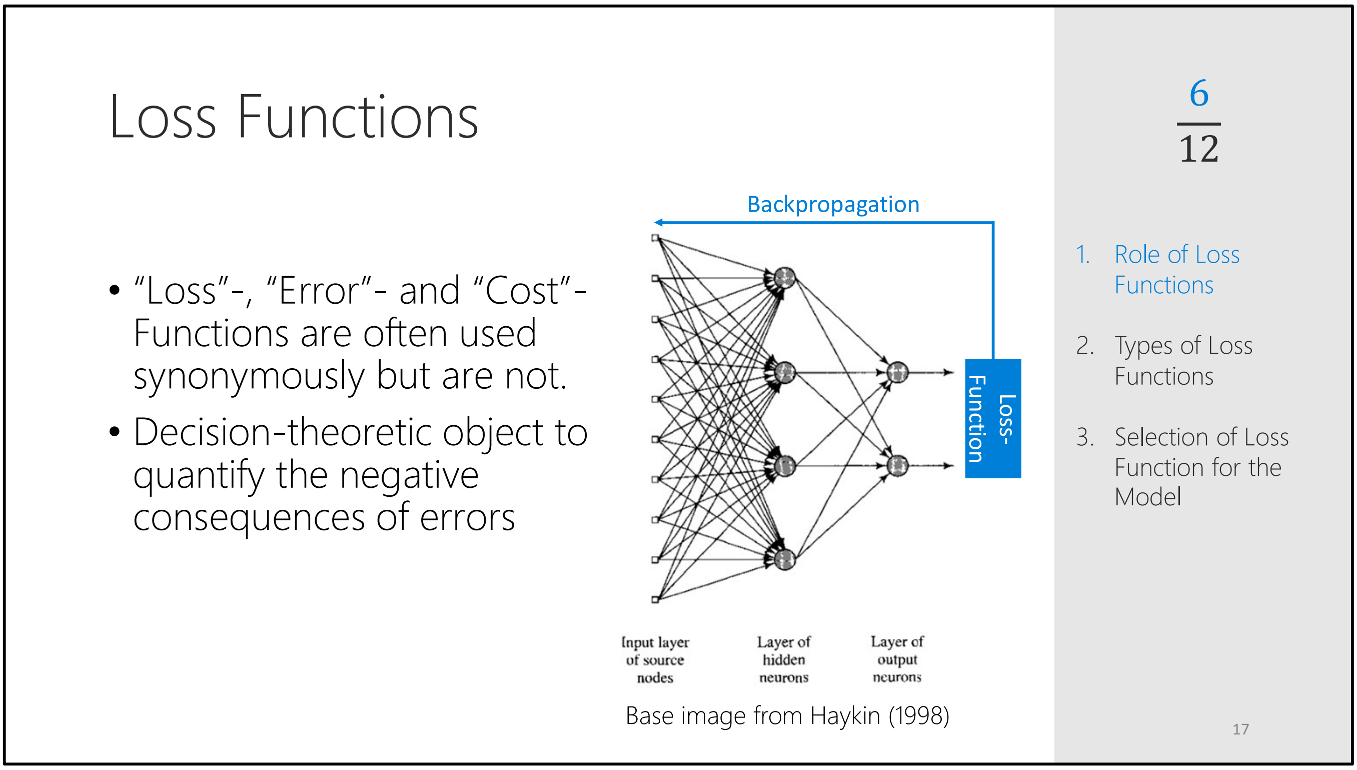
5/12

1. Role of Activation Functions
2. Types of Activation Functions
3. Choice of Activation Function in the Model

16

Gradients in neural networks are computed using back-propagation. Back-propagation calculates derivatives for each layer and propagates these derivatives from the final layer to the initial one through the chain rule (Han and Moraga, 1995). However, when there are "n" hidden layers using an activation function like the sigmoid, multiplying "n" small derivatives leads to an exponential decline in the gradient as we approach the initial layers. This diminished gradient hinders the effective updating of weights and biases during training, as seen in the red and orange functions, where the orange function represents the multiplied derivative of the sigmoid function. The ReLU function and its derivative have been chosen to address this issue as the ReLU function does not suffer from the vanishing gradient problem, making it a preferred choice for our model (Asadi and Jiang, 2020).

Additionally, the Softmax activation function converts numbers or logits into probabilities. The output of Softmax is a vector denoted as “y," which contains probabilities for each possible outcome.



Loss Functions

- "Loss"-, "Error"- and "Cost"-Functions are often used synonymously but are not.
- Decision-theoretic object to quantify the negative consequences of errors

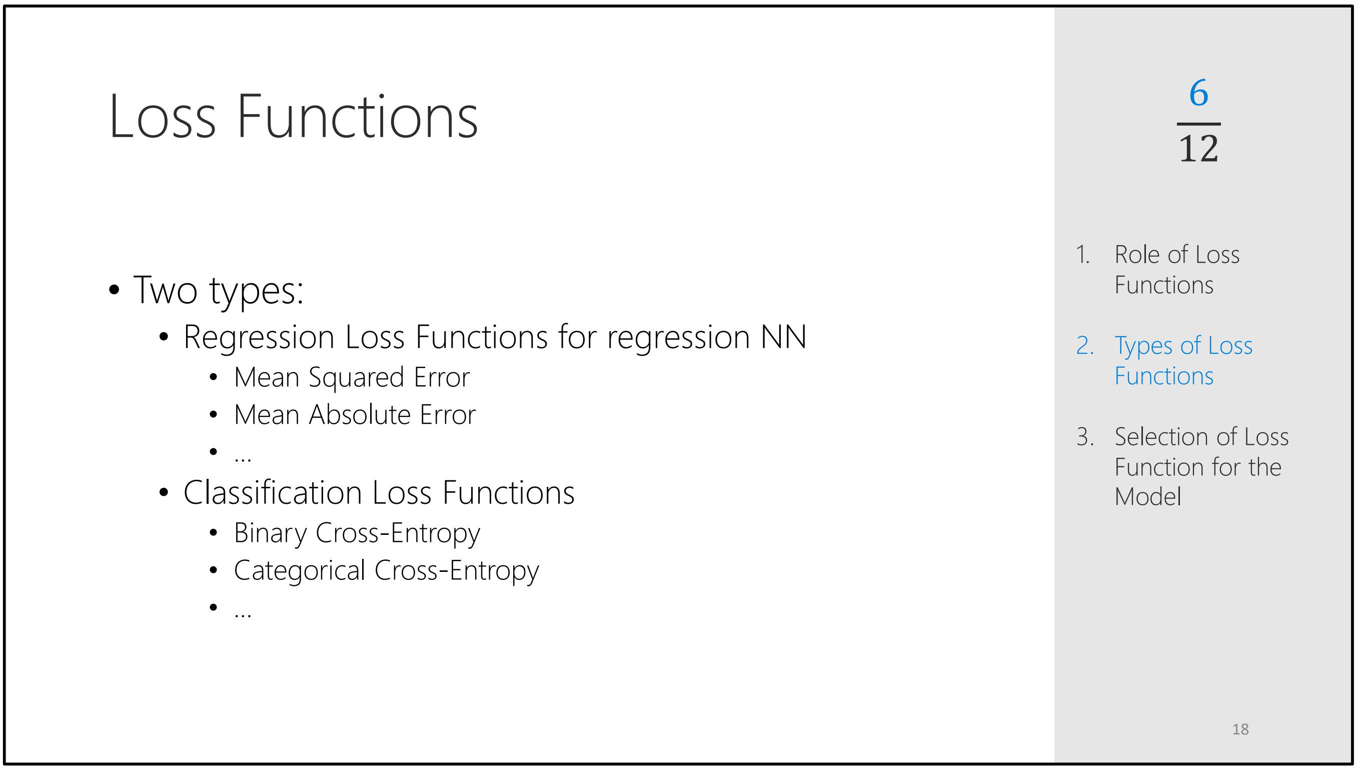
Base image from Haykin (1998)

$\frac{6}{12}$

1. Role of Loss Functions
2. Types of Loss Functions
3. Selection of Loss Function for the Model

17

The terms "Loss," "Error," and "Cost" functions are not strictly defined. In this presentation, an error function typically denotes the disparity between a prediction and an actual value. At the same time, "loss" quantifies the negative consequences of these errors from a decision-theoretic perspective. For instance, models that underestimate natural disaster predictions or environmental pollution could be problematic due to potential health implications in the latter case. Specific loss functions are employed to account for such conditions.



Loss Functions

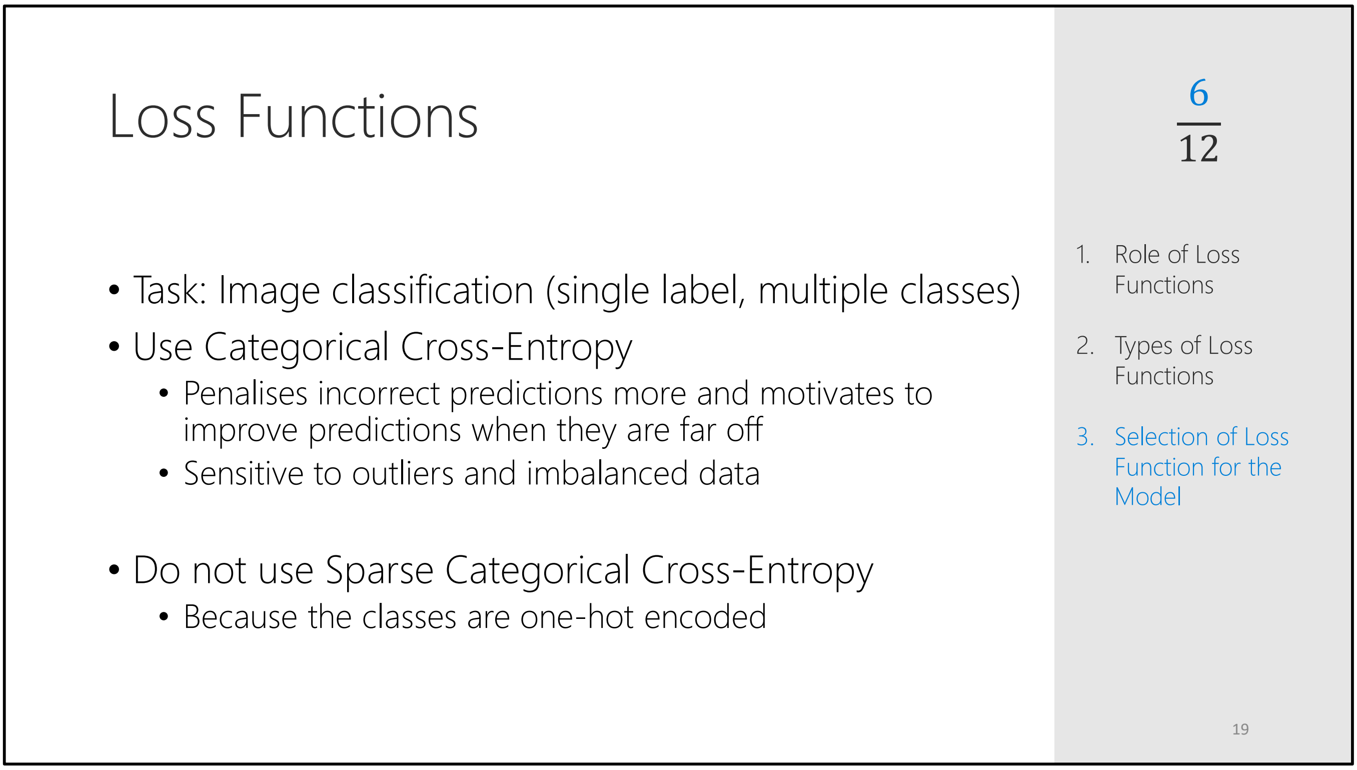
- Two types:
  - Regression Loss Functions for regression NN
    - Mean Squared Error
    - Mean Absolute Error
    - …
  - Classification Loss Functions
    - Binary Cross-Entropy
    - Categorical Cross-Entropy
    - …

$\frac{6}{12}$

1. Role of Loss Functions
2. Types of Loss Functions
3. Selection of Loss Function for the Model

18

The choice of the loss function primarily depends on the problem's complexity. In broad terms, there are distinct loss functions for regression and classification problems, each characterised by its unique mathematical framework.



Loss Functions

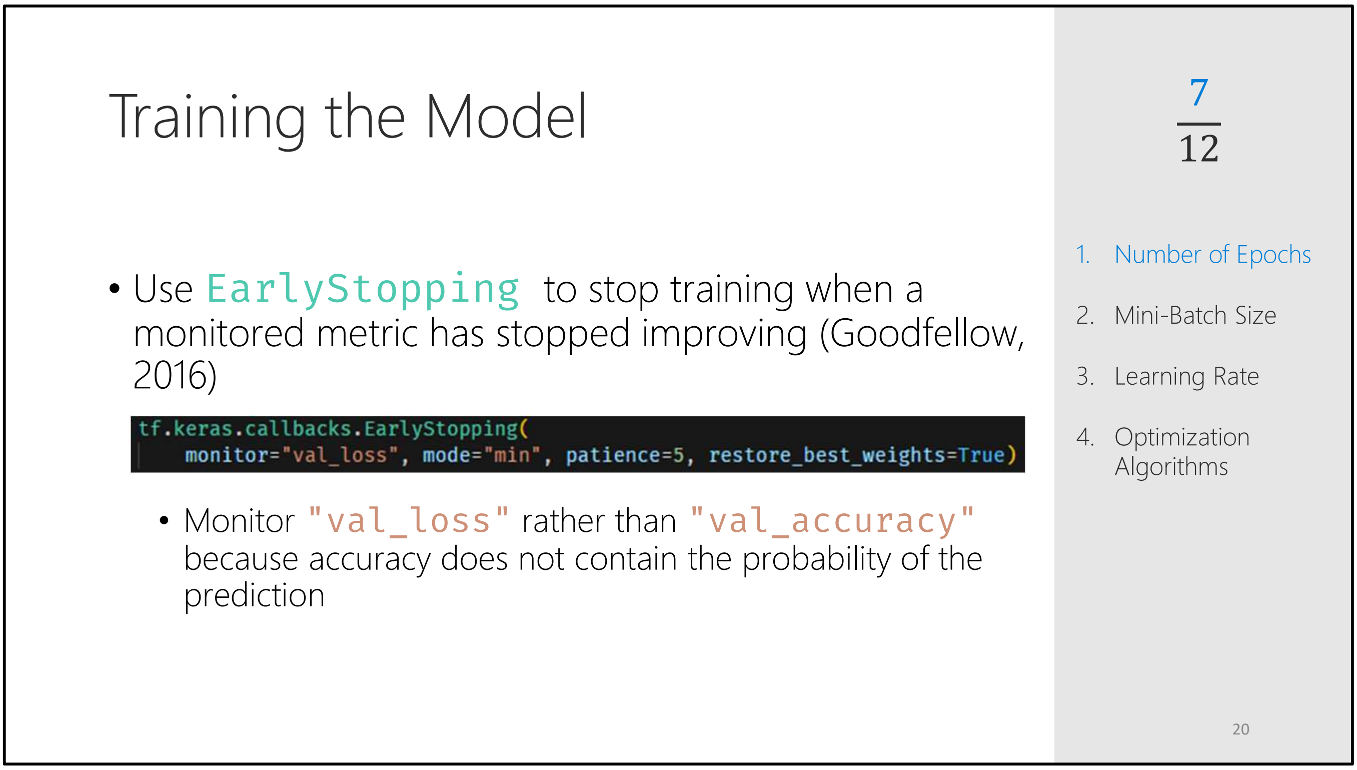
- Task: Image classification (single label, multiple classes)
- Use Categorical Cross-Entropy
  - Penalises incorrect predictions more and motivates to improve predictions when they are far off
  - Sensitive to outliers and imbalanced data
- Do not use Sparse Categorical Cross-Entropy
  - Because the classes are one-hot encoded

$\frac{6}{12}$

1. Role of Loss Functions
2. Types of Loss Functions
3. Selection of Loss Function for the Model

19

For our CIFAR-10 image classification task, which involves multiple classes, we opted for the Categorical Crossentropy loss function as Zhang and Sabuncu (2018) proposed in their recent paper. Categorical Crossentropy effectively penalises incorrect predictions, especially those significantly off the mark, thus motivating the model to enhance its predictions. It is worth noting that this loss function is sensitive to outliers and imbalanced data.



Training the Model

$\frac{7}{12}$

1. Number of Epochs
2. Mini-Batch Size
3. Learning Rate
4. Optimization Algorithms

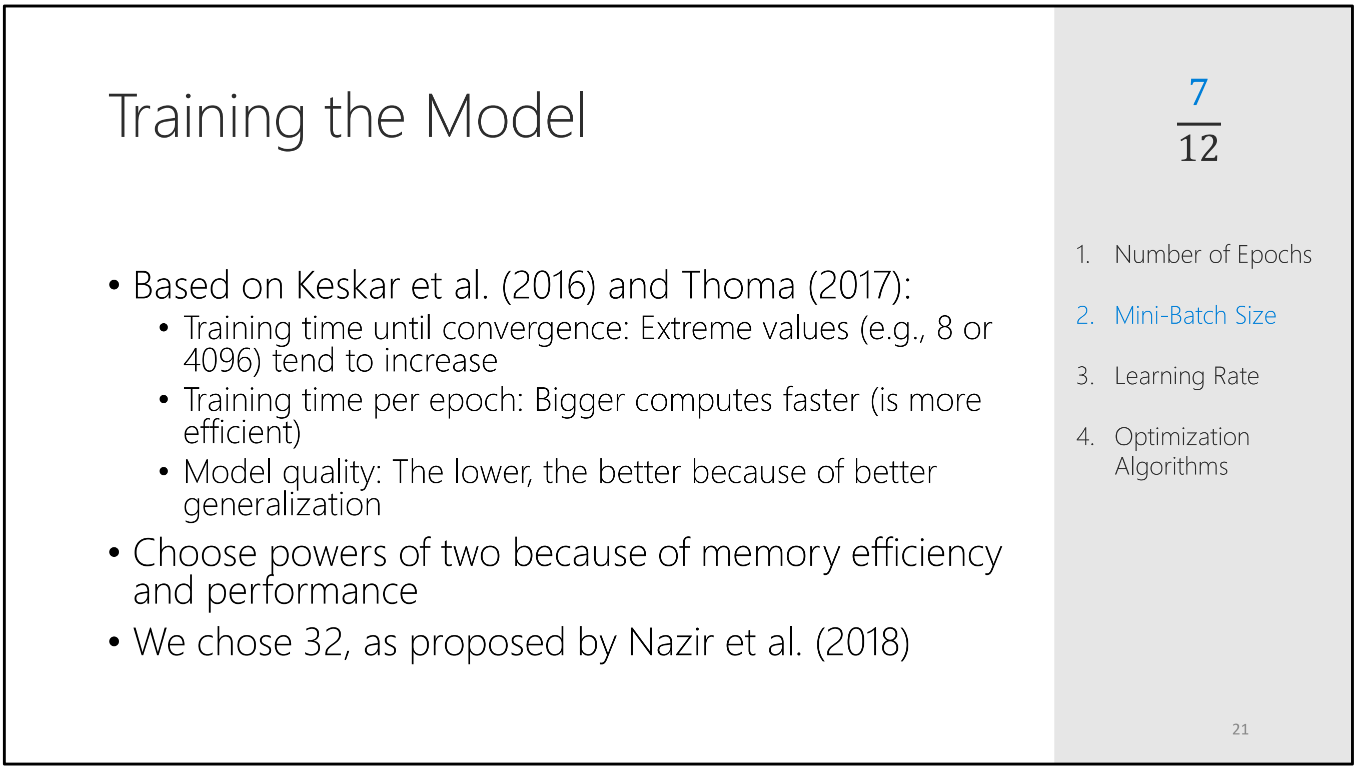
- Use EarlyStopping to stop training when a monitored metric has stopped improving (Goodfellow, 2016)

```
tf.keras.callbacks.EarlyStopping(
    monitor="val_loss", mode="min", patience=5, restore_best_weights=True)
```

  - Monitor "val_loss" rather than "val_accuracy" because accuracy does not contain the probability of the prediction

20

The early stopping strategy halts the training process after the loss fails to decrease for a set number of times (patience parameter), then reverts to the best weights in the model (Murugan and Durairaj, 2017). Dropout: a simple way to prevent neural networks from overfitting. In this instance, the rationale for optimising the loss is attributed to the probabilistic nature of its values, owing to our use of the Categorical Cross-Entropy loss function, preferring a model that confidently predicts each example correctly rather than one that correctly predicts with only a 51% confidence level.



Training the Model

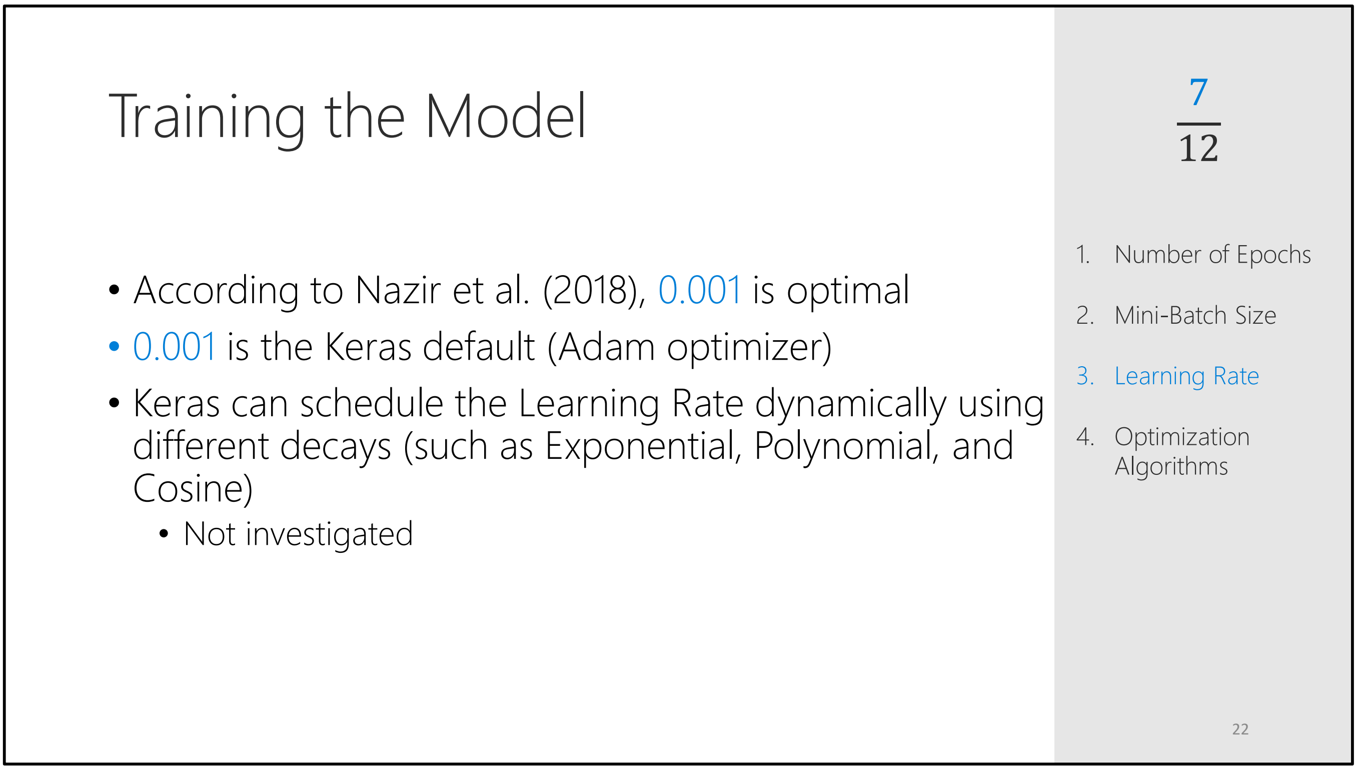
- Based on Keskar et al. (2016) and Thoma (2017):
  - Training time until convergence: Extreme values (e.g., 8 or 4096) tend to increase
  - Training time per epoch: Bigger computes faster (is more efficient)
  - Model quality: The lower, the better because of better generalization
- Choose powers of two because of memory efficiency and performance
- We chose 32, as proposed by Nazir et al. (2018)

$\frac{7}{12}$

1. Number of Epochs
2. Mini-Batch Size
3. Learning Rate
4. Optimization Algorithms

21

A batch size of 32 was used, as Keskar et al. (2016) and Thoma (2017) proposed. This choice aims to strike a balance between training efficiency, model quality, and memory efficiency, following the recommendation of Nazir et al. (2018).



Training the Model

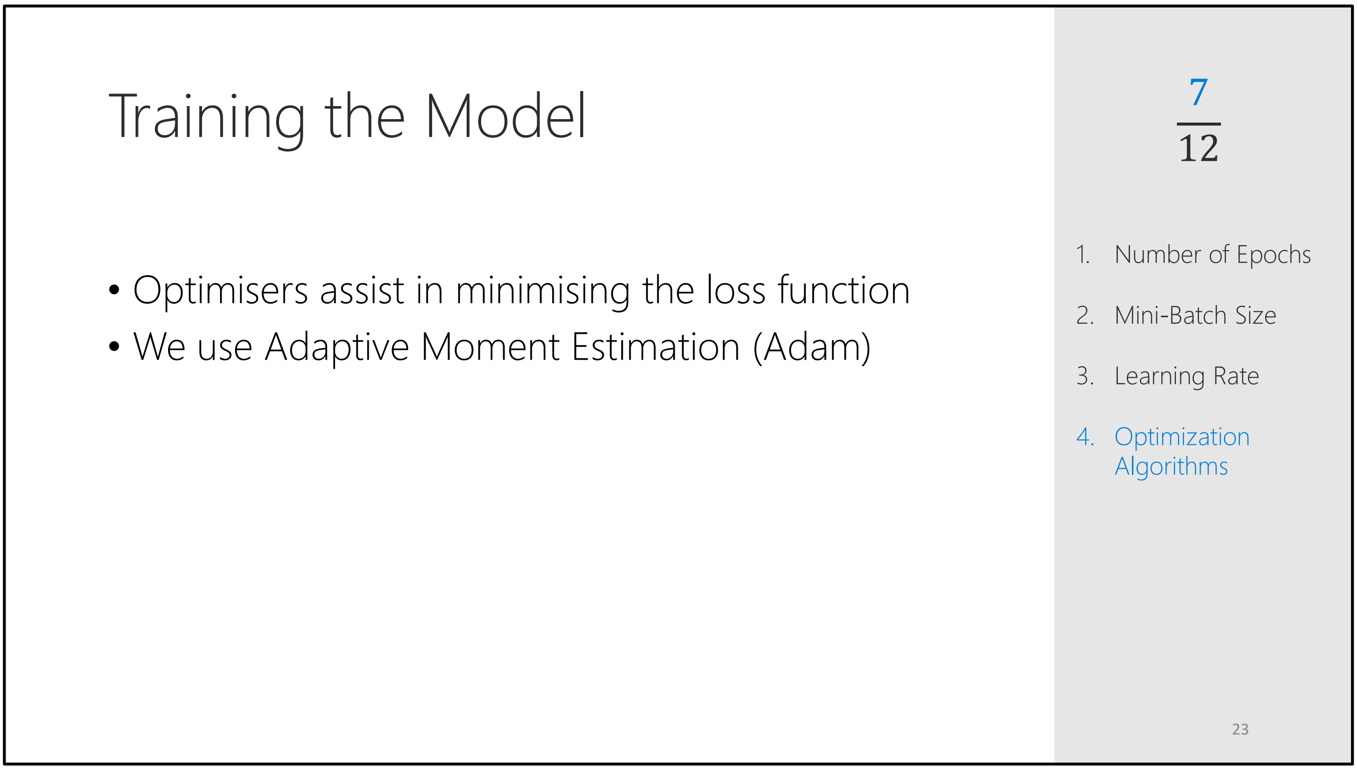
$\frac{7}{12}$

- According to Nazir et al. (2018), 0.001 is optimal
- 0.001 is the Keras default (Adam optimizer)
- Keras can schedule the Learning Rate dynamically using different decays (such as Exponential, Polynomial, and Cosine)
  - Not investigated

1. Number of Epochs
2. Mini-Batch Size
3. Learning Rate
4. Optimization Algorithms

22

In addition, a learning rate of 0.001 was selected, as suggested by Nazir et al. (2018). This value is also Keras' default for the Adam optimiser. Whilst Keras provides dynamic learning rate scheduling options, this study did not explore this option further.



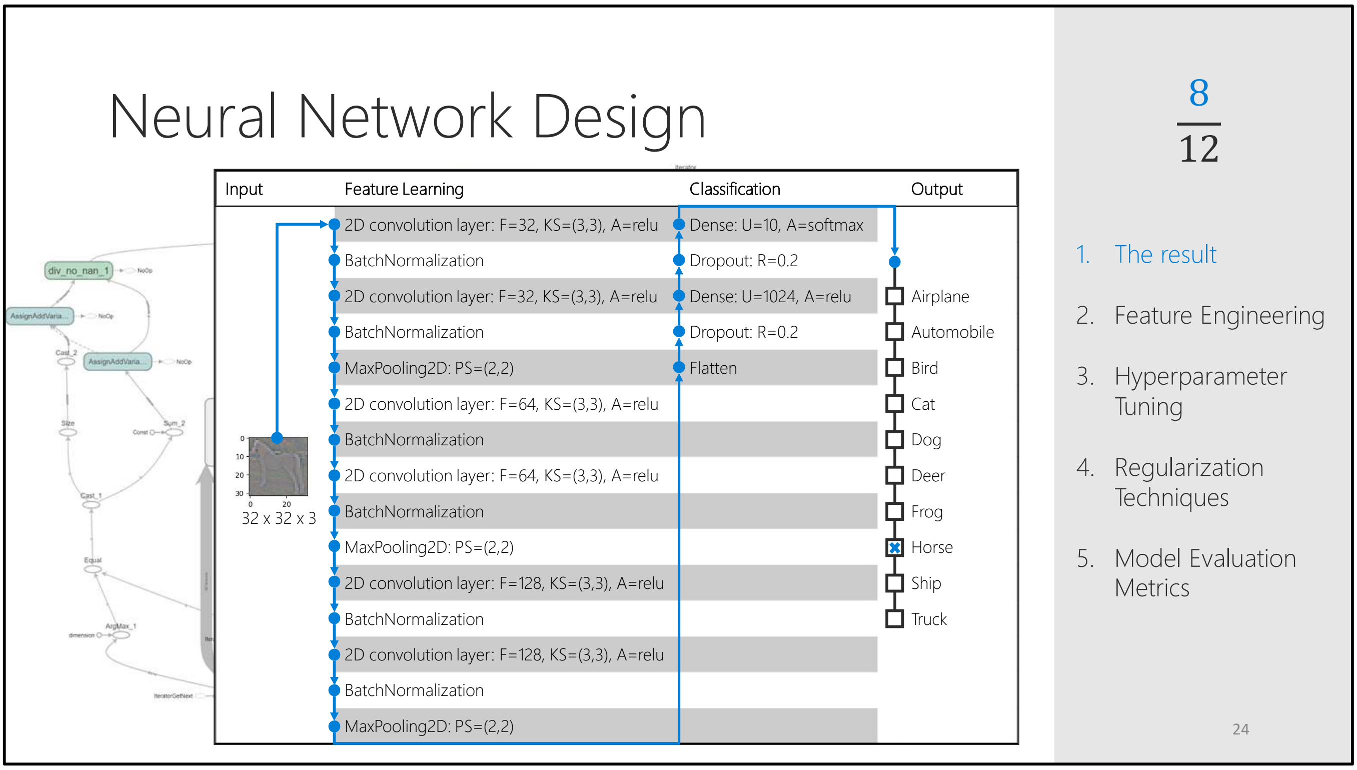
Training the Model

- Optimisers assist in minimising the loss function
- We use Adaptive Moment Estimation (Adam)

$\frac{7}{12}$

1. Number of Epochs
2. Mini-Batch Size
3. Learning Rate
4. Optimization Algorithms

The 'Adam' optimiser, short for Adaptive Moment Estimation, is a cornerstone in advanced model training (Ogundokun et al., 2022). It dynamically adjusts learning rates based on historical gradient information and combines AdaGrad and RMSProp (Zou et al., 2019), making it highly adaptive and efficient, especially for complex datasets.



Neural Network Design

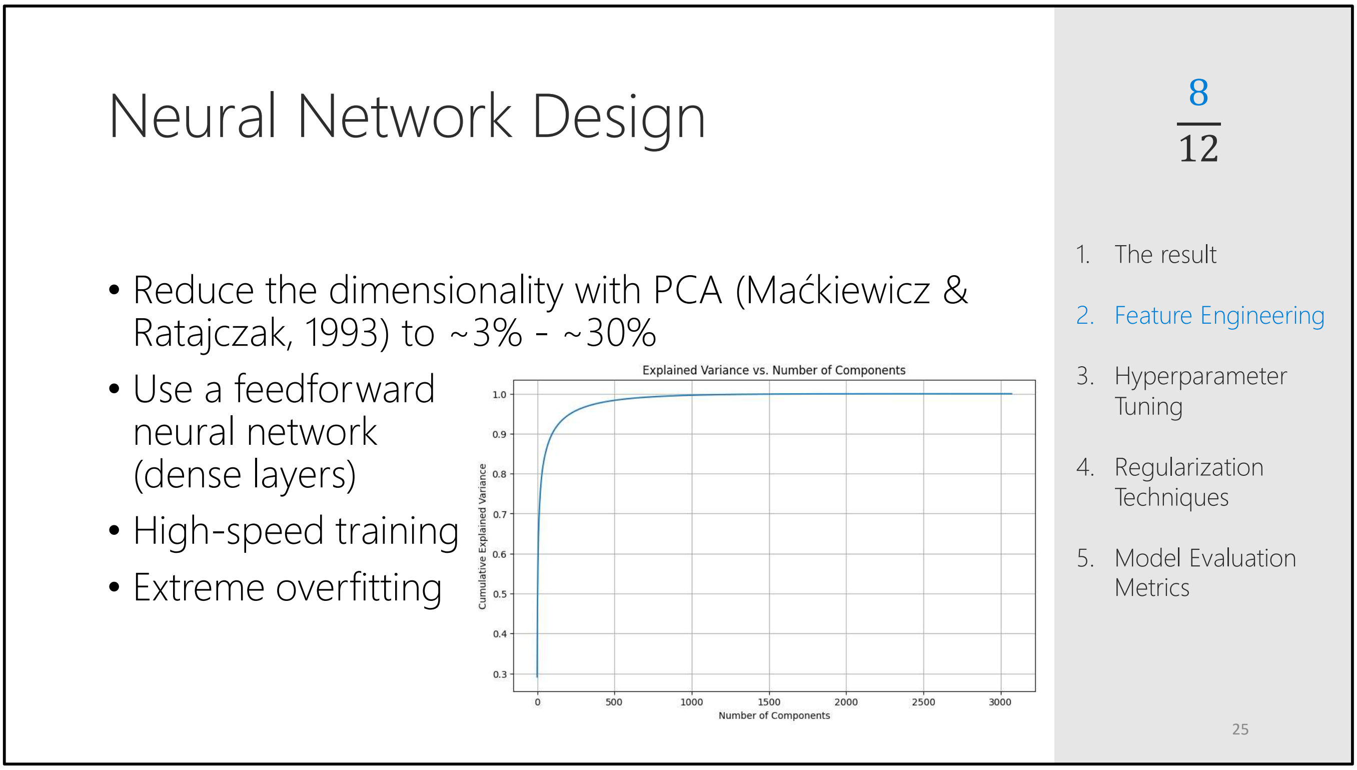
Input	Feature Learning	Classification	Output
	2D convolution layer: F=32, KS=(3,3), A=relu	Dense: U=10, A=softmax	
	BatchNormalization	Dropout: R=0.2	
	2D convolution layer: F=32, KS=(3,3), A=relu	Dense: U=1024, A=relu	□ Airplane
	BatchNormalization	Dropout: R=0.2	□ Automobile
	MaxPooling2D: PS=(2,2)	Flatten	□ Bird
	2D convolution layer: F=64, KS=(3,3), A=relu		□ Cat
	BatchNormalization		□ Dog
	2D convolution layer: F=64, KS=(3,3), A=relu		□ Deer
32 x 32 x 3	BatchNormalization		□ Frog
	MaxPooling2D: PS=(2,2)		☒ Horse
	2D convolution layer: F=128, KS=(3,3), A=relu		□ Ship
	BatchNormalization		□ Truck
	2D convolution layer: F=128, KS=(3,3), A=relu		
	BatchNormalization		
	MaxPooling2D: PS=(2,2)		

8
12

1. The result
2. Feature Engineering
3. Hyperparameter Tuning
4. Regularization Techniques
5. Model Evaluation Metrics

24

The table displayed represents the Neural Network's ultimate architecture, featuring 15 layers for feature learning and 5 for classification. It is important to note that a graphical model representation gathered through TensorBoard, serves as the background and in addition, the associated log files have also been uploaded, enabling users to launch an instance of TensorBoard and explore our model in detail.



Neural Network Design

- Reduce the dimensionality with PCA (Maćkiewicz & Ratajczak, 1993) to ~3% - ~30%
- Use a feedforward neural network (dense layers)
- High-speed training
- Extreme overfitting

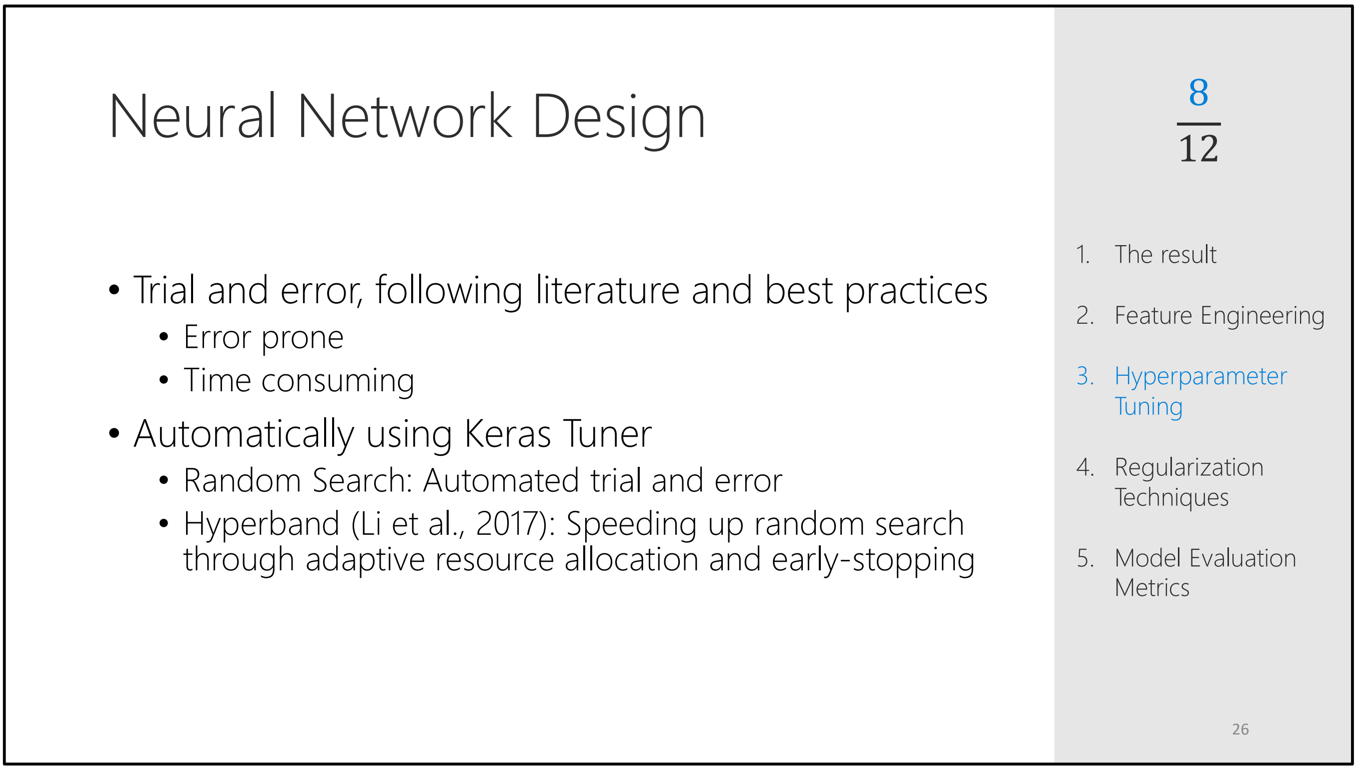
Explained Variance vs. Number of Components
Cumulative Explained Variance
Number of Components
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
500
1000
1500
2000
2500
3000

$\frac{8}{12}$

1. The result
2. Feature Engineering
3. Hyperparameter Tuning
4. Regularization Techniques
5. Model Evaluation Metrics

25

To expedite training and reduce data size, we experimented with PCA. Remarkably, our findings showed that using only 3% to 30% of the data explained 90% or more of the variance. While training with a feedforward network, we observed this method as exceptionally swift. However, the model encountered a significant challenge—severe overfitting. Training accuracy reached around 95%, but validation accuracy lagged at about 35%.



Neural Network Design

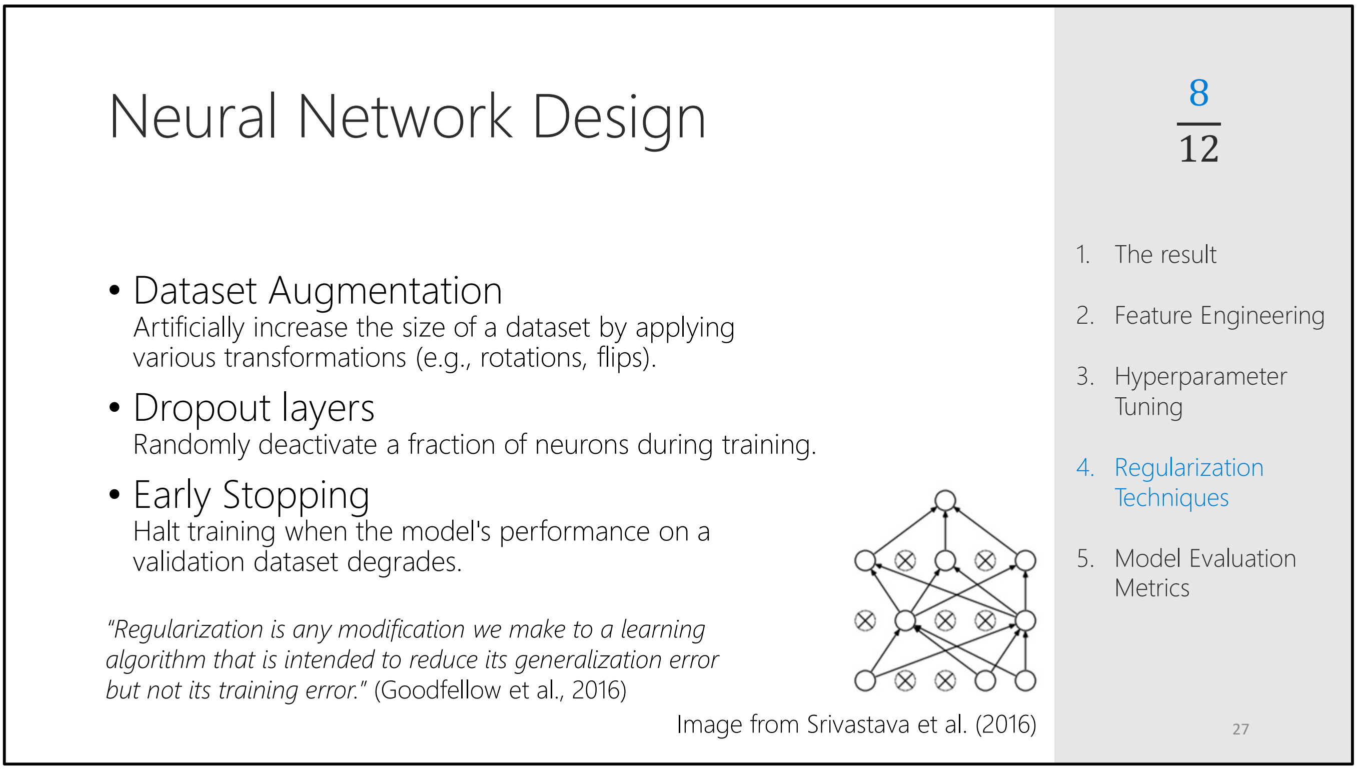
- Trial and error, following literature and best practices
  - Error prone
  - Time consuming
- Automatically using Keras Tuner
  - Random Search: Automated trial and error
  - Hyperband (Li et al., 2017): Speeding up random search through adaptive resource allocation and early-stopping

$\frac{8}{12}$

1. The result
2. Feature Engineering
3. Hyperparameter Tuning
4. Regularization Techniques
5. Model Evaluation Metrics

26

For optimal hyperparameters, the Keras Tuner was used for automated tuning. However, the 10 to 24-hour process became impractical on regular workstations, with processes cancelled and workstations going into sleep mode.



Neural Network Design

- Dataset Augmentation
  Artificially increase the size of a dataset by applying various transformations (e.g., rotations, flips).
- Dropout layers
  Randomly deactivate a fraction of neurons during training.
- Early Stopping
  Halt training when the model's performance on a validation dataset degrades.

"Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error." (Goodfellow et al., 2016)

Image from Srivastava et al. (2016)

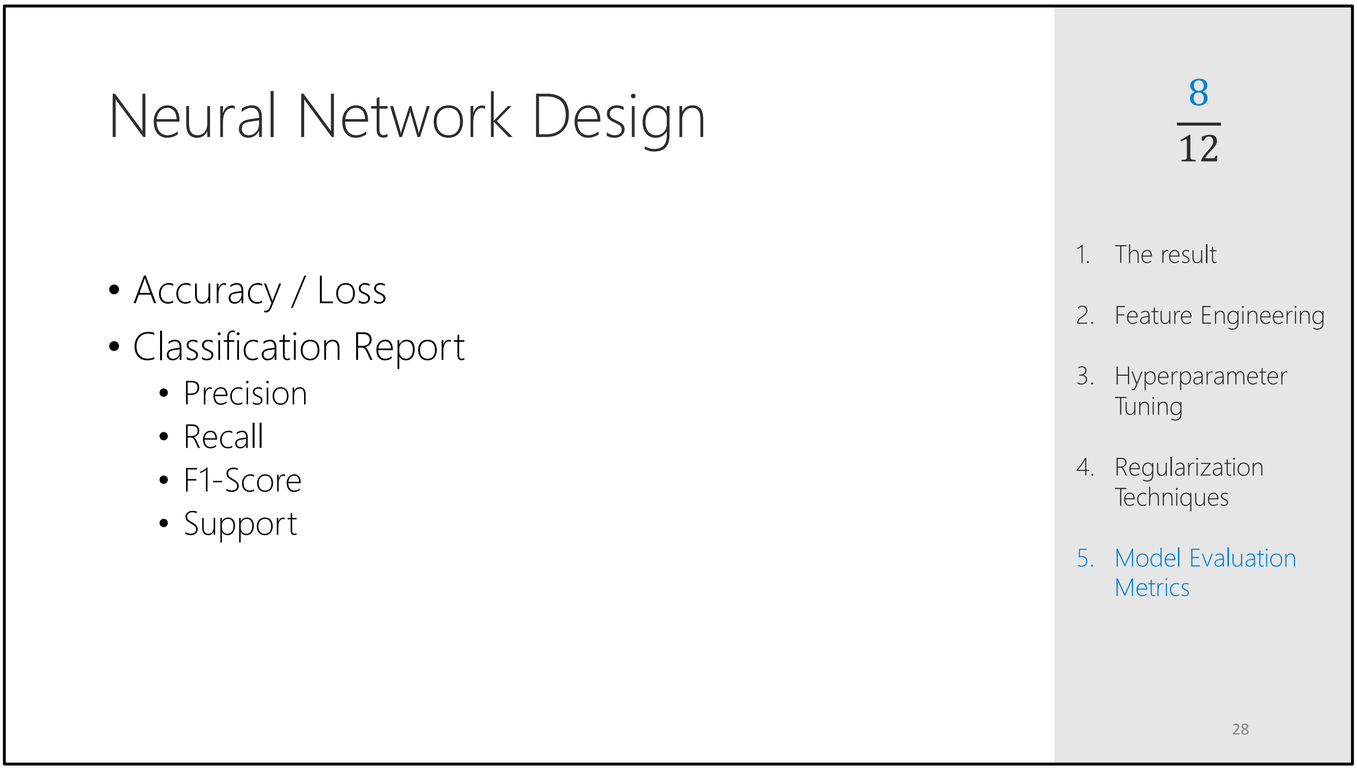
$\frac{8}{12}$

1. The result
2. Feature Engineering
3. Hyperparameter Tuning
4. Regularization Techniques
5. Model Evaluation Metrics

27

Various methodologies were instrumental in increasing the efficacy of the model. Dataset Augmentation, Dropout Layers, and Early Stopping were incorporated into the approach. Perronnin et al. (2012) argued that Dataset Augmentation amplified the training dataset by introducing diversified variations, thereby augmenting generalisation and performance.

As proposed by Srivastava et al. (2016), dropout layers were used as an overfitting mitigation mechanism by intermittently terminating connections among neurons.



Neural Network Design

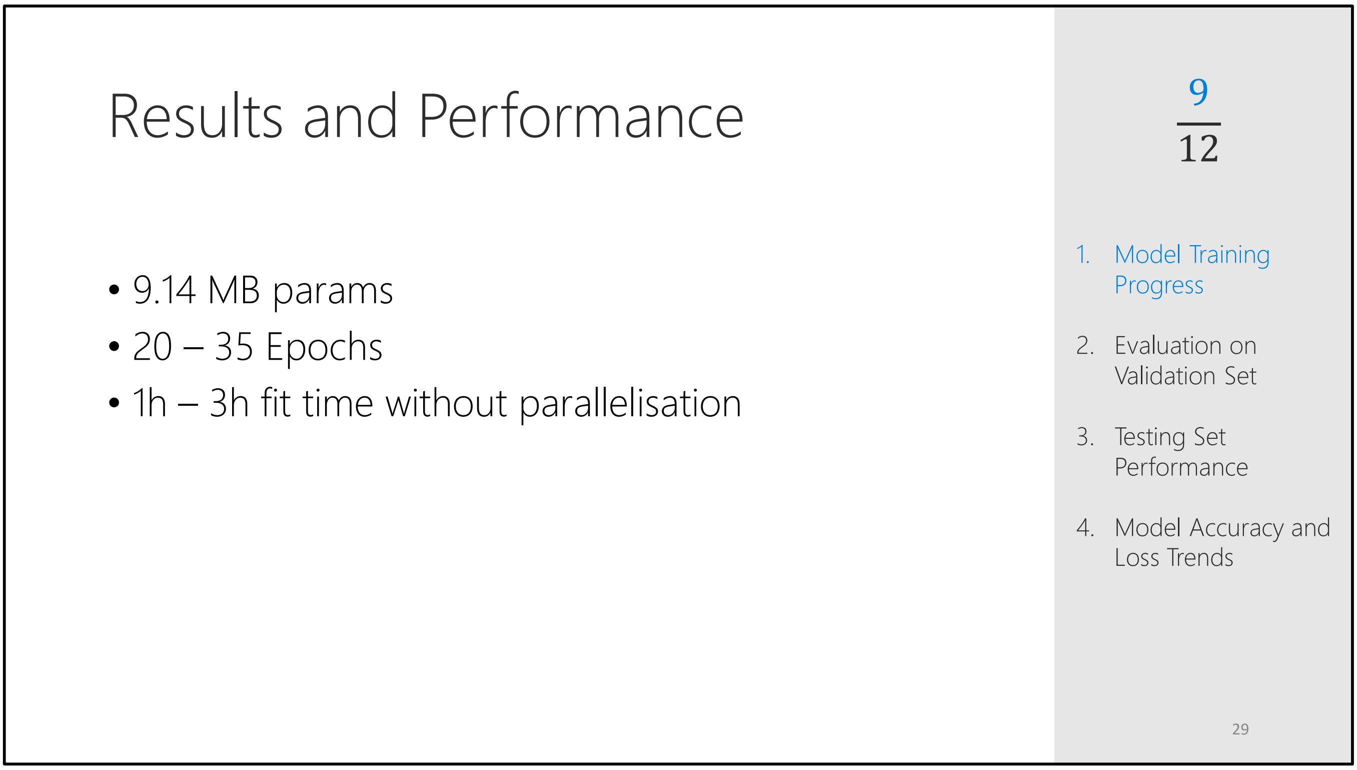
- Accuracy / Loss
- Classification Report
  - Precision
  - Recall
  - F1-Score
  - Support

$\frac{8}{12}$

1. The result
2. Feature Engineering
3. Hyperparameter Tuning
4. Regularization Techniques
5. Model Evaluation Metrics

28

In our model evaluation, various essential metrics are considered, such as accuracy, which gauges the model's proficiency in correctly classifying instances. Furthermore, the evaluation includes loss measurement through Categorical Cross-Entropy to quantify the disparity between predicted probabilities and actual values. Additionally, precision is assessed to determine the model's effectiveness in reducing false positive predictions within the positive category. Similarly, recall measures the model's capability to identify all true positive instances. The F1-Score is also evaluated, balancing precision and recall in binary classification scenarios. Lastly, support is provided to offer contextual information, indicating the actual occurrences for each class in a classification problem.



Results and Performance

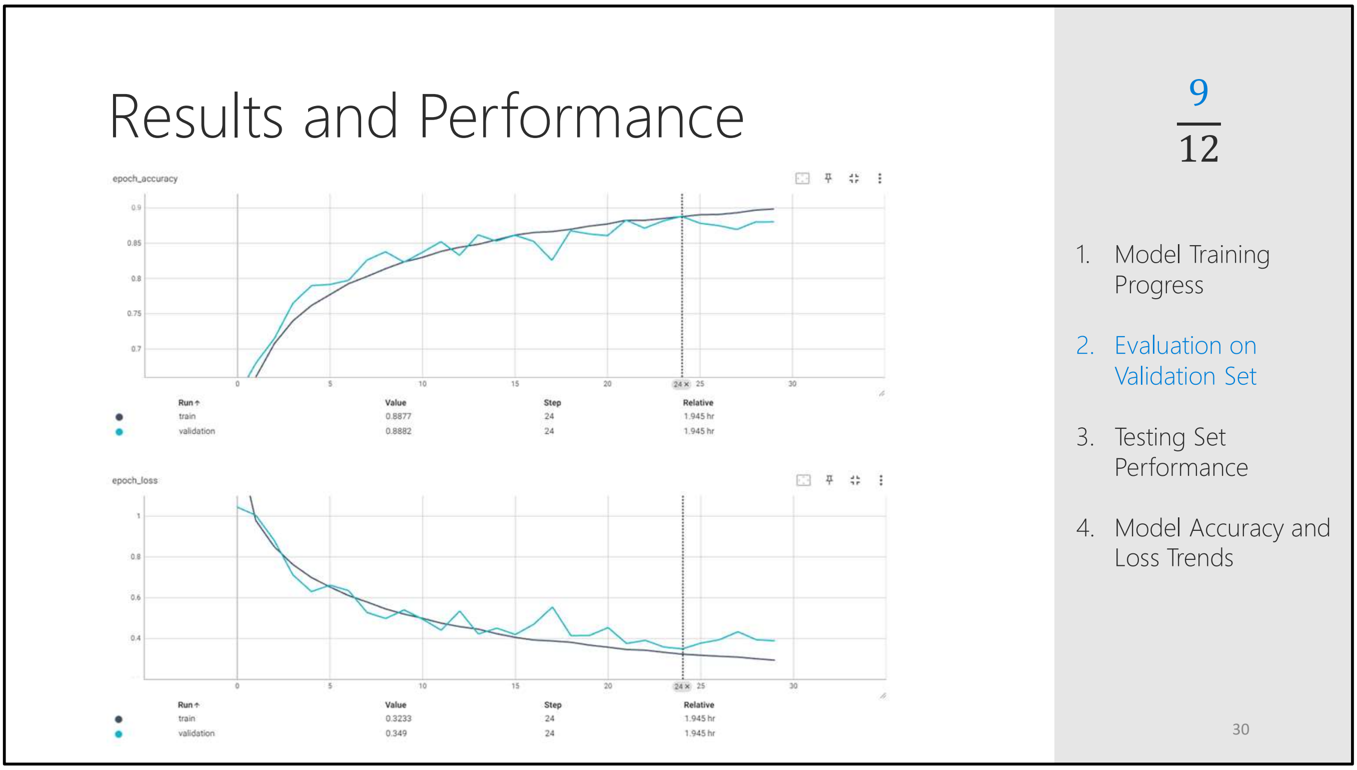
- 9.14 MB params
- 20 – 35 Epochs
- 1h – 3h fit time without parallelisation

$\frac{9}{12}$

1. Model Training Progress
2. Evaluation on Validation Set
3. Testing Set Performance
4. Model Accuracy and Loss Trends

29

The following section delves into the achieved results. Training typically spanned 20 to 35 epochs and took anywhere from one to three hours, depending on the workstation used. For demonstration purposes we added a time-lapse of the training process.



Results and Performance

	Run ↑	Value	Step	Relative
●	train	0.8877	24	1.945 hr
●	validation	0.8882	24	1.945 hr

	Run ↑	Value	Step	Relative
●	train	0.3233	24	1.945 hr
●	validation	0.349	24	1.945 hr

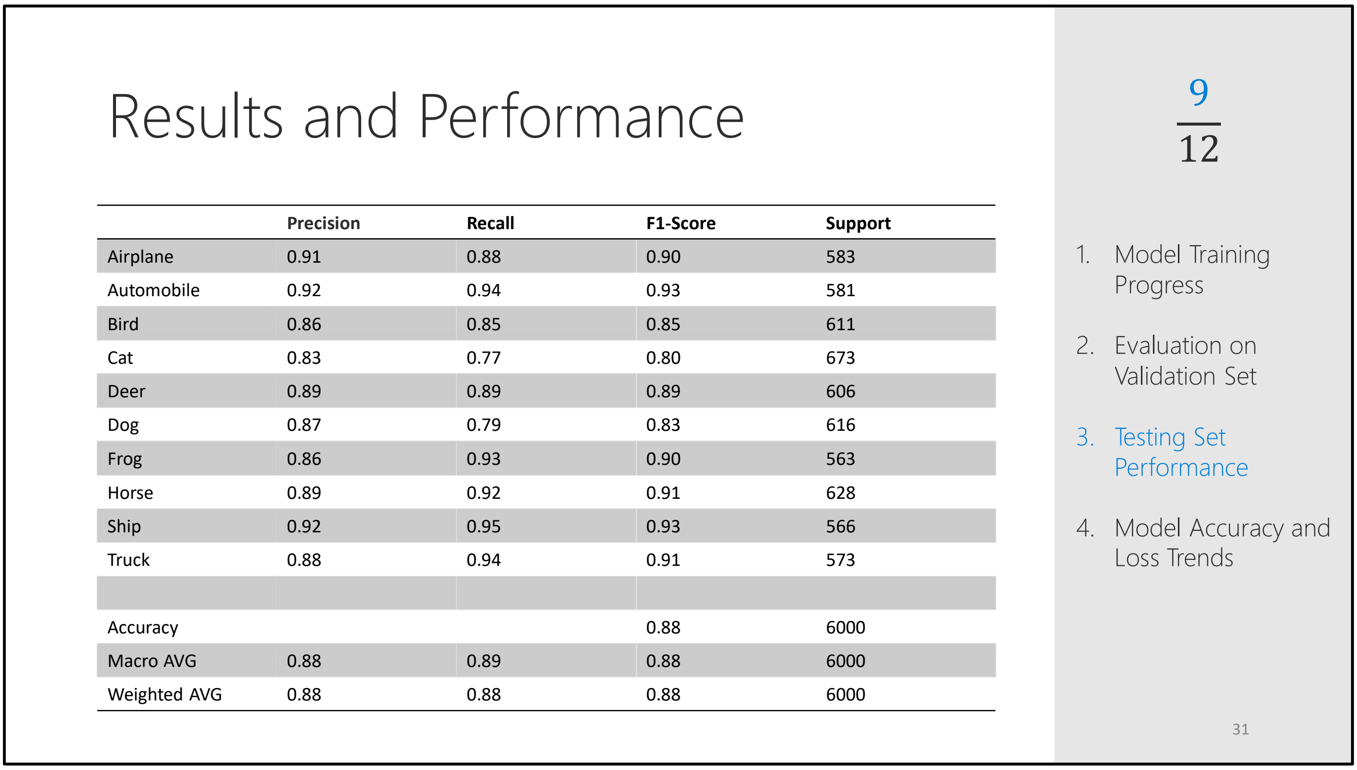
$\frac{9}{12}$

1. Model Training Progress
2. Evaluation on Validation Set
3. Testing Set Performance
4. Model Accuracy and Loss Trends

30

Our model exhibits slightly better performance on the training set, an outcome in the training process, and there are no signs of overfitting.

The epoch accuracy and epoch loss diagrams were extracted from the TensorBoard application. Notably, the vertical slider at epoch 24 indicates the moment when the loss value on the validation set began to rise again. This situation prompted the early stopping callback to halt the training process.



Results and Performance

	Precision	Recall	F1-Score	Support
Airplane	0.91	0.88	0.90	583
Automobile	0.92	0.94	0.93	581
Bird	0.86	0.85	0.85	611
Cat	0.83	0.77	0.80	673
Deer	0.89	0.89	0.89	606
Dog	0.87	0.79	0.83	616
Frog	0.86	0.93	0.90	563
Horse	0.89	0.92	0.91	628
Ship	0.92	0.95	0.93	566
Truck	0.88	0.94	0.91	573
Accuracy			0.88	6000
Macro AVG	0.88	0.89	0.88	6000
Weighted AVG	0.88	0.88	0.88	6000

9

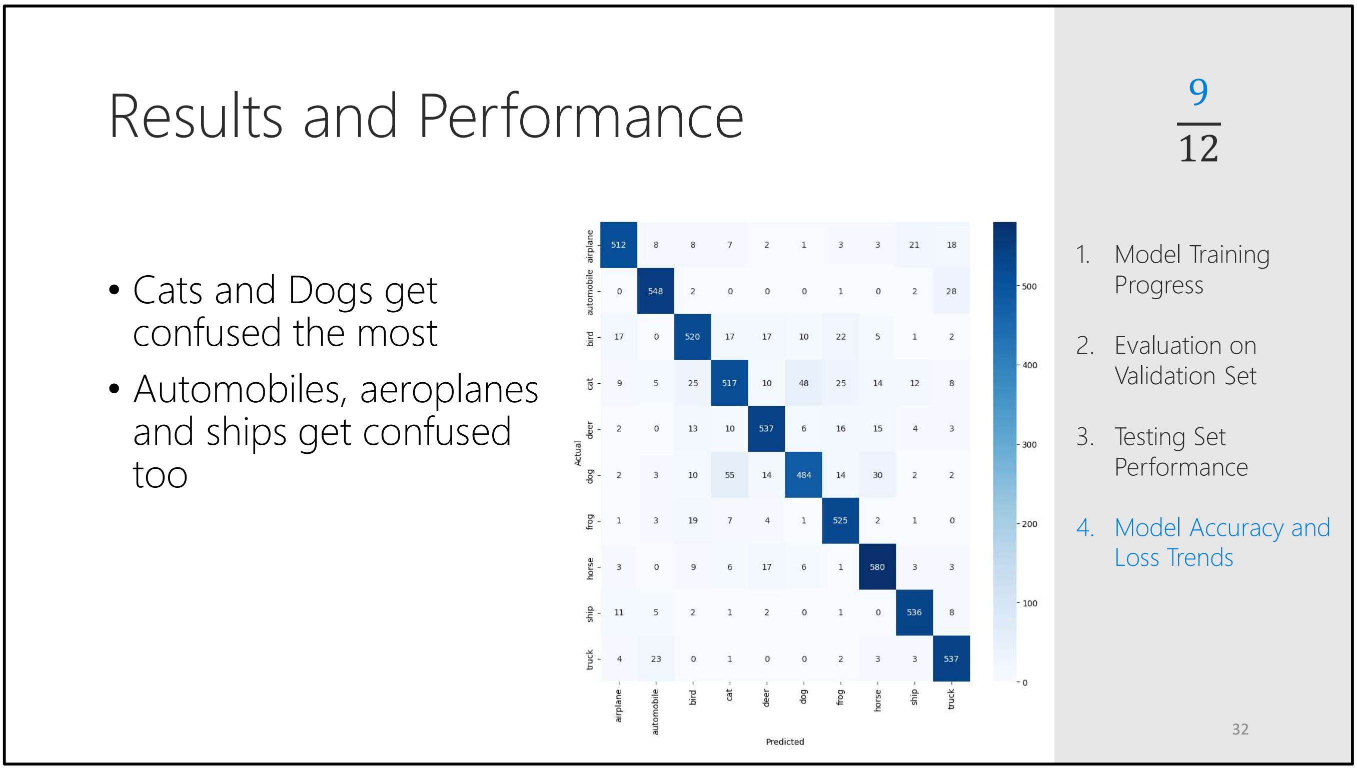
12

1. Model Training Progress
2. Evaluation on Validation Set
3. Testing Set Performance
4. Model Accuracy and Loss Trends

31

Upon employing Scikit-Learn's classification_report to evaluate performance on the testing set, commendable precision, recall, and F1-Score were observed, all approaching 90%.

Further examination by category reveals that Automobiles and Ships achieved the highest scores, while Cats and Dogs performed less favourably.



Results and Performance

- Cats and Dogs get confused the most
- Automobiles, aeroplanes and ships get confused too

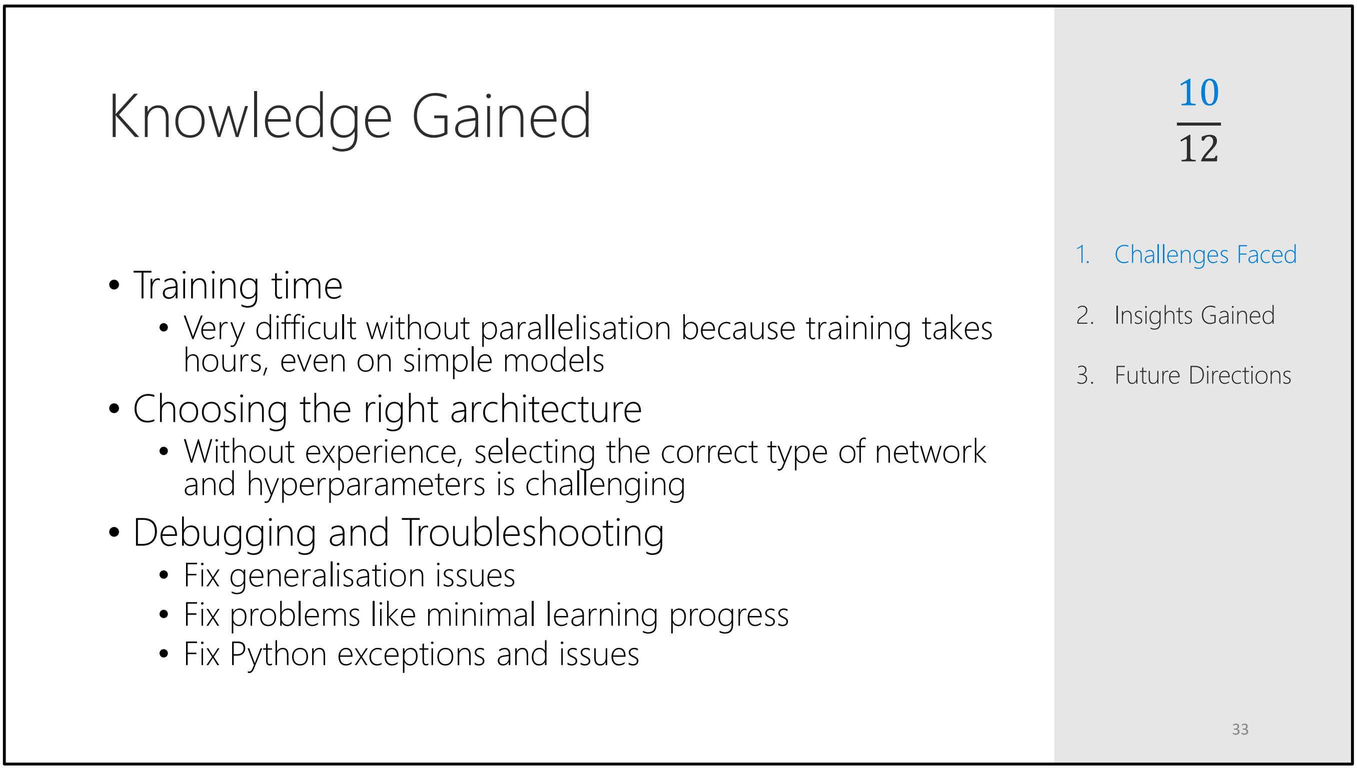
Actual \ Predicted	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	512	8	8	7	2	1	3	3	21	18
automobile	0	548	2	0	0	0	1	0	2	28
bird	17	0	520	17	17	10	22	5	1	2
cat	9	5	25	517	10	48	25	14	12	8
deer	2	0	13	10	537	6	16	15	4	3
dog	2	3	10	55	14	484	14	30	2	2
frog	1	3	19	7	4	1	525	2	1	0
horse	3	0	9	6	17	6	1	580	3	3
ship	11	5	2	1	2	0	1	0	536	8
truck	4	23	0	1	0	0	2	3	3	537

$\frac{9}{12}$

1. Model Training Progress
2. Evaluation on Validation Set
3. Testing Set Performance
4. Model Accuracy and Loss Trends

The predictive confusion matrix highlights that Cats and Dogs (and vice versa) are the most frequently confused categories. Additionally, there are notable confusions within the Automobiles, Aeroplanes , and Ships group.

Similar observations can be observed in other papers addressing the CIFAR-10 classification challenge (Aslam and Nassif (2023); Jiang and Goldsztein, 2023; Lv, 2020)



Knowledge Gained

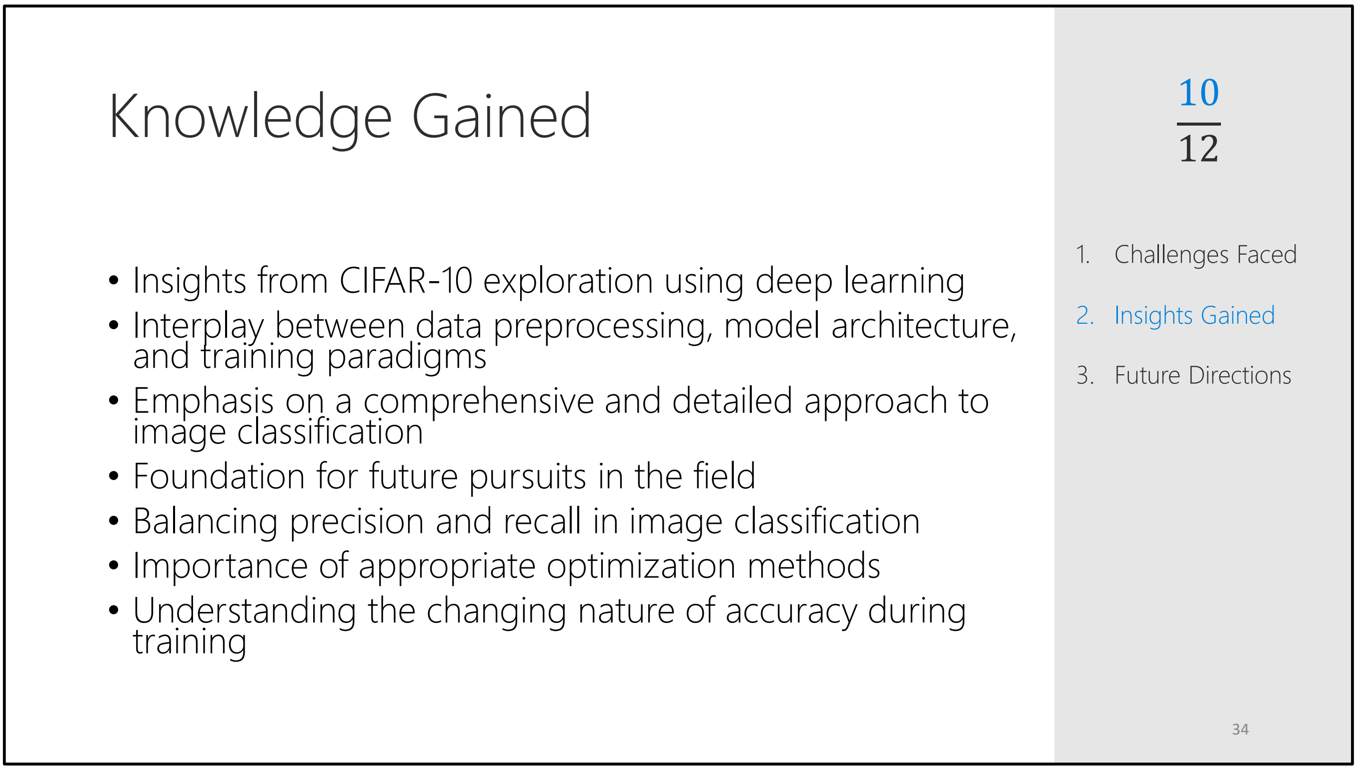
$\frac{10}{12}$

1. Challenges Faced
2. Insights Gained
3. Future Directions

- Training time
  - Very difficult without parallelisation because training takes hours, even on simple models
- Choosing the right architecture
  - Without experience, selecting the correct type of network and hyperparameters is challenging
- Debugging and Troubleshooting
  - Fix generalisation issues
  - Fix problems like minimal learning progress
  - Fix Python exceptions and issues

33

Whilst engaging in deep learning endeavours, challenges emerge. Training time is a primary concern, necessitating parallelisation due to the time-intensive nature (Shallue, 2018). Selecting the right architecture and tuning hyperparameters is challenging for the inexperienced. Debugging involves addressing generalisation, learning progress, and Python issues, highlighting the complexity of deep learning tasks.



Knowledge Gained

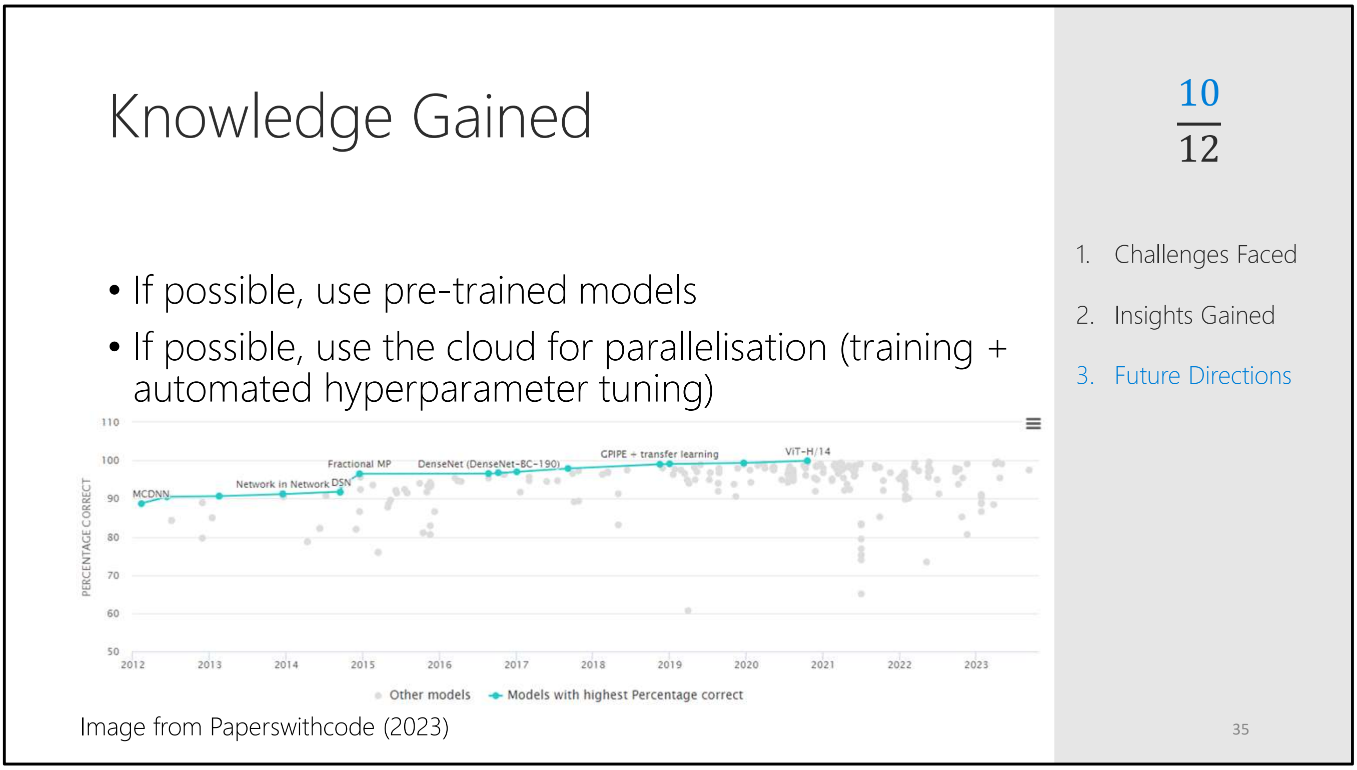
- Insights from CIFAR-10 exploration using deep learning
- Interplay between data preprocessing, model architecture, and training paradigms
- Emphasis on a comprehensive and detailed approach to image classification
- Foundation for future pursuits in the field
- Balancing precision and recall in image classification
- Importance of appropriate optimization methods
- Understanding the changing nature of accuracy during training

$\frac{10}{12}$

1. Challenges Faced
2. Insights Gained
3. Future Directions

34

Throughout the CIFAR-10 exploration using deep learning, numerous insights were gained. The intricate interplay between data pre-processing, model architecture, and training paradigms emphasised the essential need for a comprehensive and detailed approach to image classification. Notwithstanding, the insights derived from the model's strengths and areas for improvement lay a solid groundwork for future pursuits in this field. These findings underscore the delicate equilibrium between precision and recall, highlight the importance of appropriate optimisation methods, and shed light on the changing nature of accuracy as training progresses.



Knowledge Gained

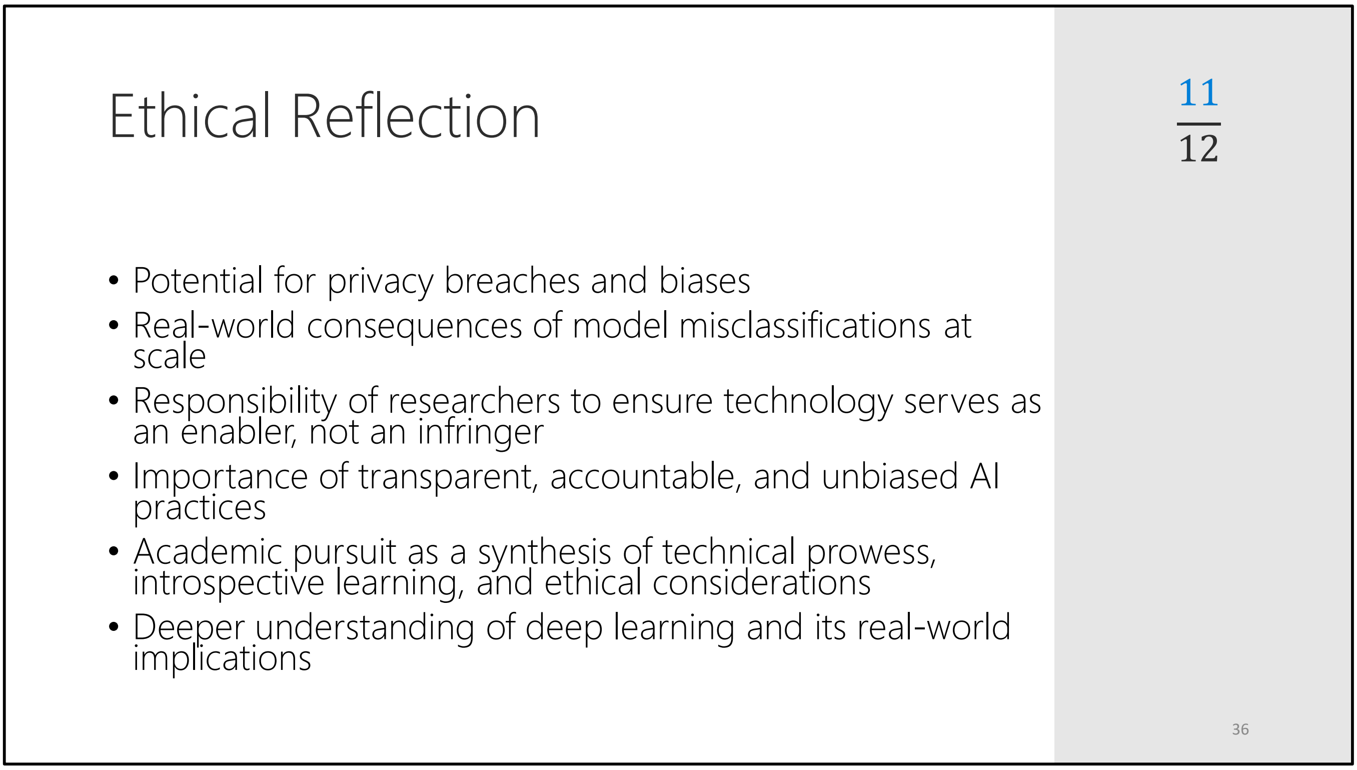
$\frac{10}{12}$

1. Challenges Faced
2. Insights Gained
3. Future Directions

- If possible, use pre-trained models
- If possible, use the cloud for parallelisation (training + automated hyperparameter tuning)

Image from Paperswithcode (2023)

The achieved results on the CIFAR-10 dataset align with the performance levels of 2012, as evident from Paperswithcode (2023). Presently, pre-trained models such as VIT-H/14 can achieve an impressive 99.9% accuracy, showing that in particular contexts, the use of prebuilt models is a highly favoured approach.



Ethical Reflection

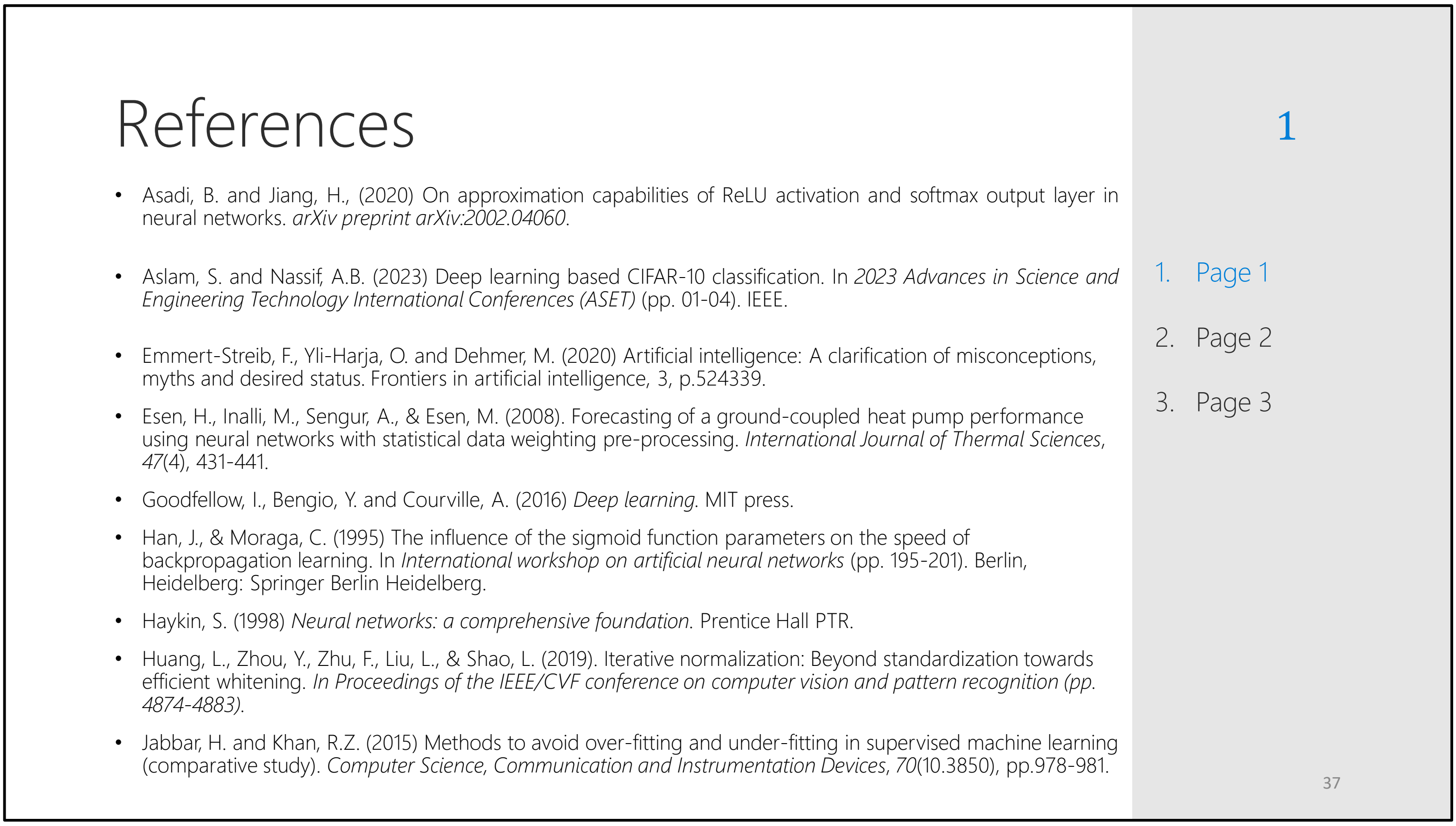
$\frac{11}{12}$

- Potential for privacy breaches and biases
- Real-world consequences of model misclassifications at scale
- Responsibility of researchers to ensure technology serves as an enabler, not an infringer
- Importance of transparent, accountable, and unbiased AI practices
- Academic pursuit as a synthesis of technical prowess, introspective learning, and ethical considerations
- Deeper understanding of deep learning and its real-world implications

36

Ethically, while the achievements with the CIFAR-10 dataset are commendable, they also necessitate contemplation of broader implications. If not wielded judiciously, image recognition can inadvertently breach privacy norms or perpetuate biases (Van Noorden, 2020). Though seemingly benign in a controlled dataset, the model's occasional misclassifications could have real-world ramifications when applied at scale. Moreover, with increasing advancements in deep learning, there is an onus on researchers to ensure that technology serves as an enabler, not an infringer, of rights. Ensuring transparent, accountable, and unbiased AI practices becomes paramount.

In essence, this exercise has been more than an academic pursuit; it has been a synthesis of technical prowess, introspective learning, and ethical considerations, culminating in a deeper understanding of the vast landscape of deep learning and its implications in the real world.

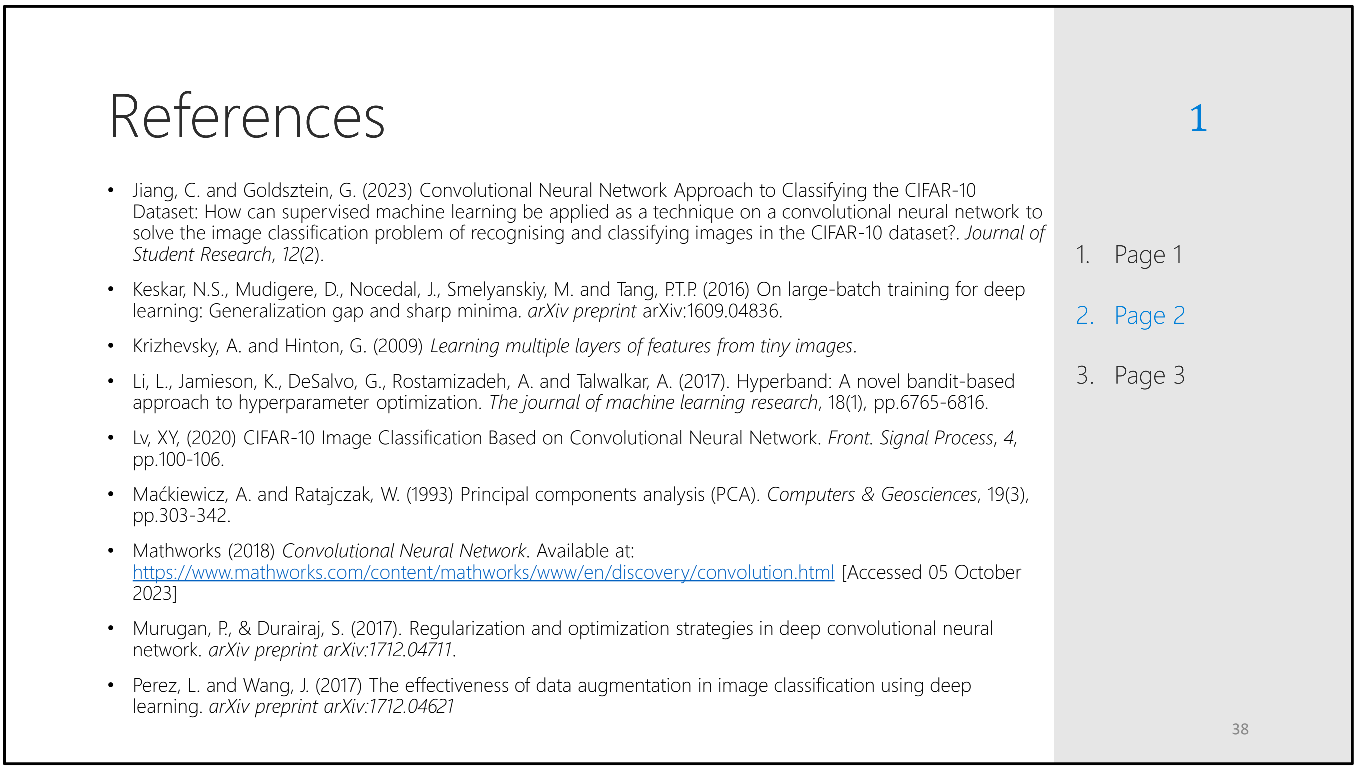


References

- Asadi, B. and Jiang, H., (2020) On approximation capabilities of ReLU activation and softmax output layer in neural networks. *arXiv preprint arXiv:2002.04060*.
- Aslam, S. and Nassif, A.B. (2023) Deep learning based CIFAR-10 classification. In *2023 Advances in Science and Engineering Technology International Conferences (ASET)* (pp. 01-04). IEEE.
- Emmert-Streib, F., Yli-Harja, O. and Dehmer, M. (2020) Artificial intelligence: A clarification of misconceptions, myths and desired status. Frontiers in artificial intelligence, 3, p.524339.
- Esen, H., Inalli, M., Sengur, A., & Esen, M. (2008). Forecasting of a ground-coupled heat pump performance using neural networks with statistical data weighting pre-processing. *International Journal of Thermal Sciences*, *47*(4), 431-441.
- Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep learning*. MIT press.
- Han, J., & Moraga, C. (1995) The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks* (pp. 195-201). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Haykin, S. (1998) *Neural networks: a comprehensive foundation*. Prentice Hall PTR.
- Huang, L., Zhou, Y., Zhu, F., Liu, L., & Shao, L. (2019). Iterative normalization: Beyond standardization towards efficient whitening. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 4874-4883).*
- Jabbar, H. and Khan, R.Z. (2015) Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). *Computer Science, Communication and Instrumentation Devices*, *70*(10.3850), pp.978-981.

1

1. Page 1
2. Page 2
3. Page 3



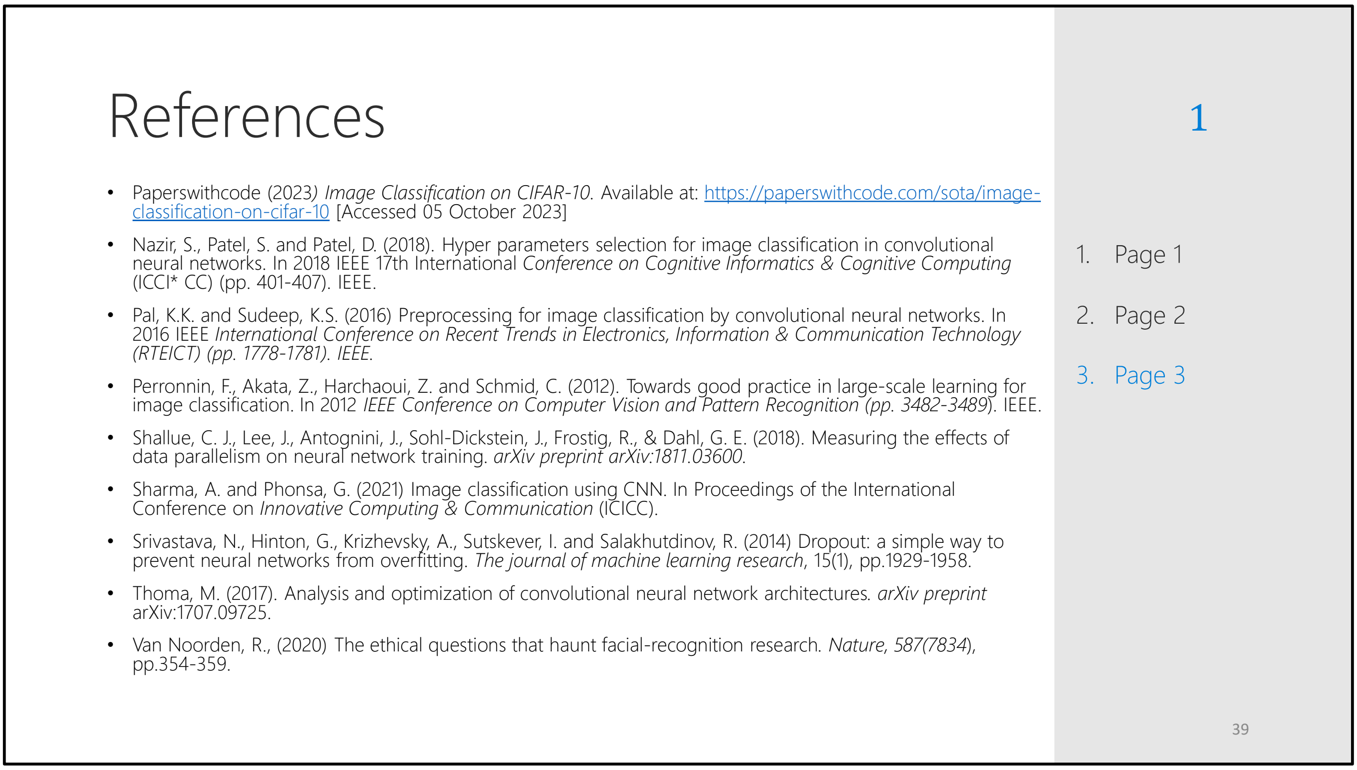
References

- Jiang, C. and Goldsztein, G. (2023) Convolutional Neural Network Approach to Classifying the CIFAR-10 Dataset: How can supervised machine learning be applied as a technique on a convolutional neural network to solve the image classification problem of recognising and classifying images in the CIFAR-10 dataset?. *Journal of Student Research*, *12*(2).
- Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M. and Tang, P.T.P. (2016) On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint* arXiv:1609.04836.
- Krizhevsky, A. and Hinton, G. (2009) *Learning multiple layers of features from tiny images*.
- Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. and Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The journal of machine learning research*, 18(1), pp.6765-6816.
- Lv, XY, (2020) CIFAR-10 Image Classification Based on Convolutional Neural Network. *Front. Signal Process*, *4*, pp.100-106.
- Maćkiewicz, A. and Ratajczak, W. (1993) Principal components analysis (PCA). *Computers & Geosciences*, 19(3), pp.303-342.
- Mathworks (2018) *Convolutional Neural Network*. Available at: https://www.mathworks.com/content/mathworks/www/en/discovery/convolution.html [Accessed 05 October 2023]
- Murugan, P., & Durairaj, S. (2017). Regularization and optimization strategies in deep convolutional neural network. *arXiv preprint arXiv:1712.04711*.
- Perez, L. and Wang, J. (2017) The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*

1

1. Page 1
2. Page 2
3. Page 3

38

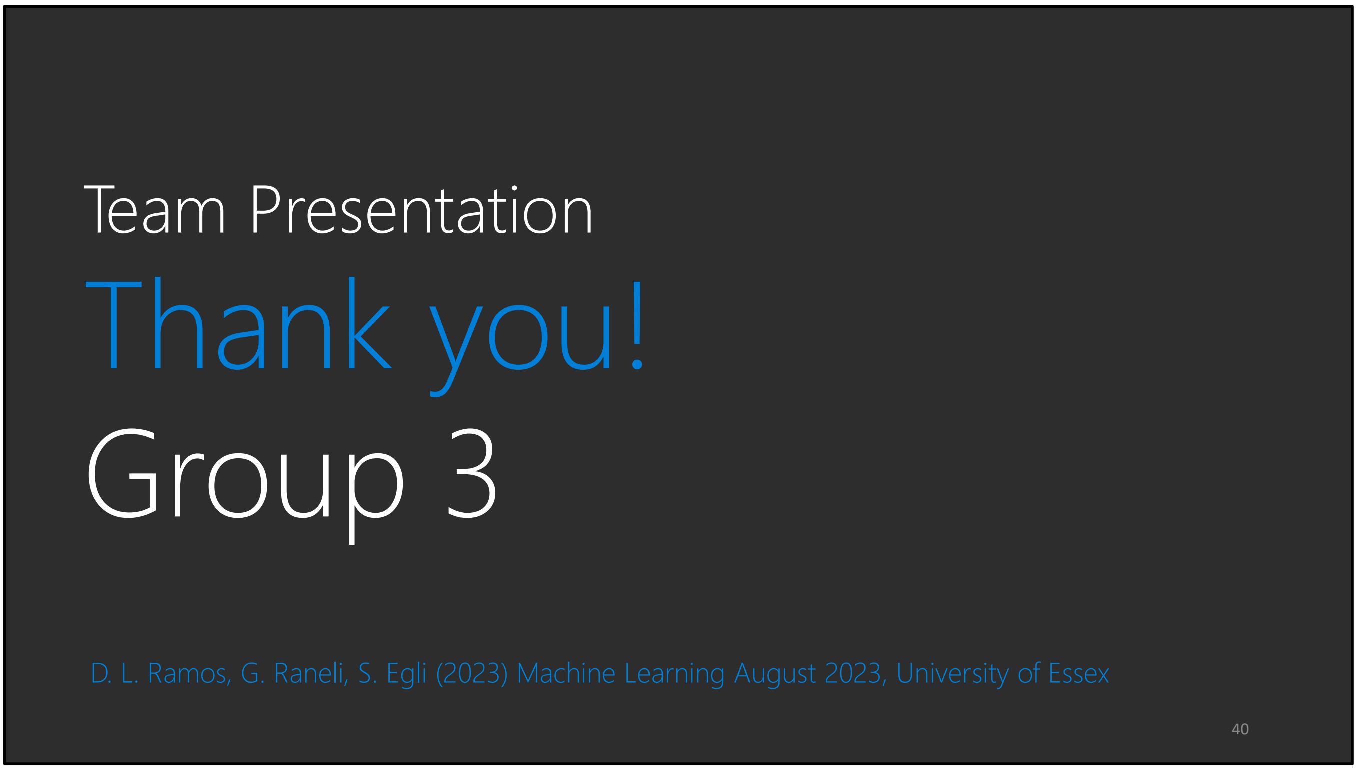


References

- Paperswithcode (2023*) Image Classification on CIFAR-10.* Available at: https://paperswithcode.com/sota/image-classification-on-cifar-10 [Accessed 05 October 2023]
- Nazir, S., Patel, S. and Patel, D. (2018). Hyper parameters selection for image classification in convolutional neural networks. In 2018 IEEE 17th International *Conference on Cognitive Informatics & Cognitive Computing* (ICCI* CC) (pp. 401-407). IEEE.
- Pal, K.K. and Sudeep, K.S. (2016) Preprocessing for image classification by convolutional neural networks. In 2016 IEEE *International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT) (pp. 1778-1781). IEEE.*
- Perronnin, F., Akata, Z., Harchaoui, Z. and Schmid, C. (2012). Towards good practice in large-scale learning for image classification. In 2012 *IEEE Conference on Computer Vision and Pattern Recognition (pp. 3482-3489*). IEEE.
- Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., & Dahl, G. E. (2018). Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600.*
- Sharma, A. and Phonsa, G. (2021) Image classification using CNN. In Proceedings of the International Conference on *Innovative Computing & Communication* (ICICC).
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), pp.1929-1958.
- Thoma, M. (2017). Analysis and optimization of convolutional neural network architectures. *arXiv preprint* arXiv:1707.09725.
- Van Noorden, R., (2020) The ethical questions that haunt facial-recognition research. *Nature, 587(7834*), pp.354-359.

1

1. Page 1
2. Page 2
3. Page 3



Team Presentation

Thank you!

Group 3

D. L. Ramos, G. Raneli, S. Egli (2023) Machine Learning August 2023, University of Essex

Thank you very much for your attention.