Fault Diagnosis of Asynchronous Motor through Thermal Imaging using Convolutional Neural Networks

Vision and Cognitive Systems Project

June 28th 2024

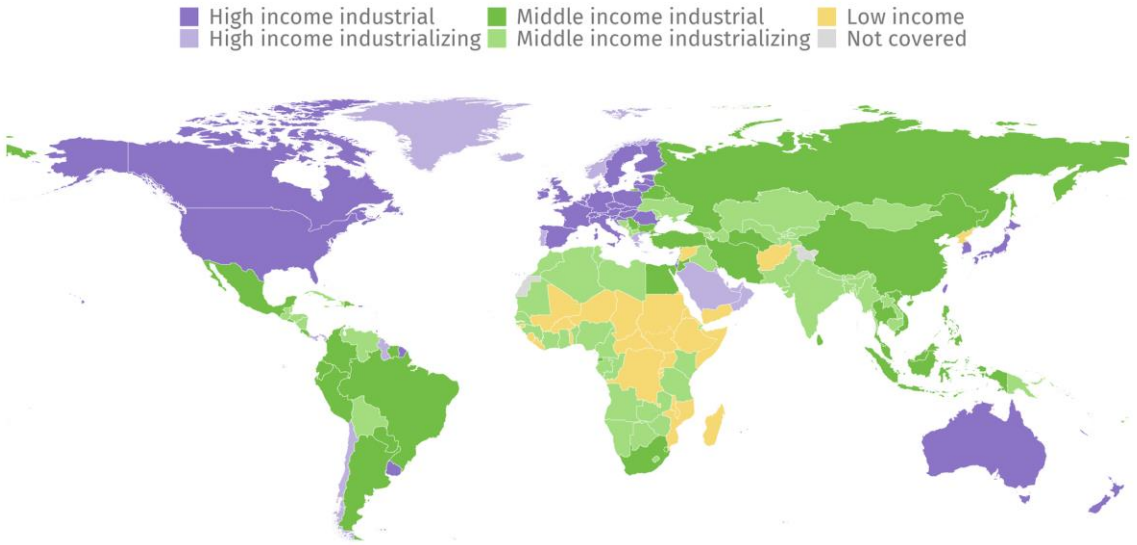
Alejandra Cruces
Mario Tapia

Contents

Introduction

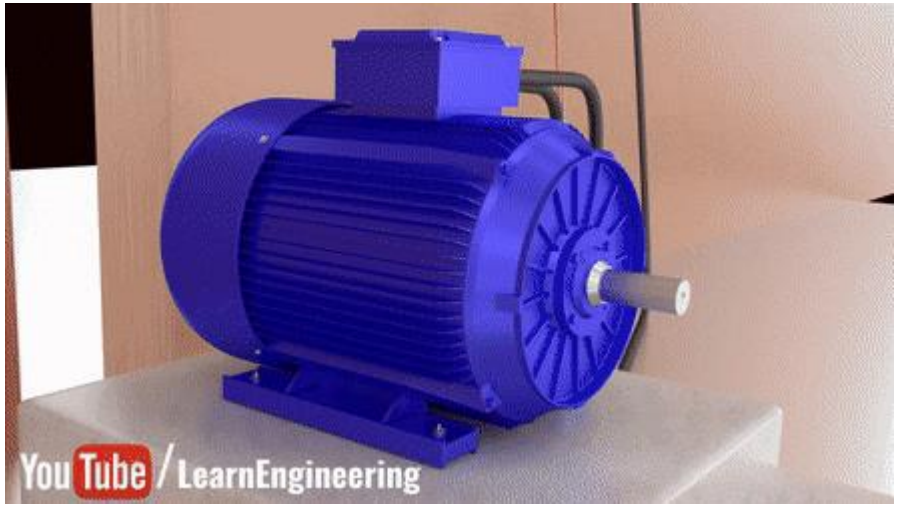
Introduction: Energetic context

- Electric motors are responsible for about 2/3 of industrial power consumption in industrially developed nations
- They power a variety of equipment including wind blowers, water pumps, compressors
- The three-phase induction motor is the most commonly used type due to its reliability and low cost



Introduction: What is an electric motor?

- An electric motor is a device that transforms electrical energy into mechanical energy
- It mainly consists of a static part (stator), and a rotating part (rotor)
- We focus on the three-phase induction (asynchrounous) motor



Introduction: What is an electric motor?

- An electric motor is a device that transforms electrical energy into mechanical energy
- It mainly consists of a static part (stator), and a rotating part (rotor)
- We focus on the three-phase induction (asynchrounous) motor

Introduction: Types of failures

- **Stator electrical** faults which are produced by stator winding open or short circuited
- **Rotor electrical** faults produced by broken bar(s) or cracked end-ring for squirrel-cage machines
- **Rotor mechanical** faults such as bearing damage, eccentricity, bent shaft, misalignment, locked rotor, and fan failure
- Failure of power electronic components of the drive system

Introduction: Types of failures

- **Stator** (**electrical**) faults which are defined by stator winding open or **short circuited**
- **Rotor electrical** faults broken bar(s) or cracked end-ring for squirrel-cage machines
- **Rotor mechanical** faults such as bearing damage, eccentricity, bent shaft, misalignment, **locked rotor, and fan failure**
- Failure of power electronic components of the drive system

Introduction: Types of failures

- Each type of fault generates characteristic features in the machine's behavior
  - ✓ Infrared Thermography

Introduction: Benefits of early detection

- ✓ **Cost savings:** Estimating potential failures before they occur
- ✓ **Preventive maintenance:** Facilitate pre-planned preventive machine schedules
- ✓ **Production continuity:** Prevents unexpected stop in the production lines
- ✓ **Efficiency improvement:** Improve the induction motor efficiency

Related work

Related Work

Najafi et al. (2020)

Multi-step Model:

"Hot" and "Cold" classification

Segmentation: Identify potential Regions of Interest (faults).

Feature Extraction: First- and second-order statistics

Classification: Decision Tree classifier

Training-Test Split: 50%-50%

Validation: 10-fold cross-validation

Results:

Accuracy: 93.8%

Sakalli et al. (2020)

Transfer Learning with ResNet50

Two test methods: 70%-30% and 80%-20% training-test splits

Hyperparameter tunning

Results:

Accuracy: 100%

Training-Test Split: 80%-20% split yields better results

Low learning rate provides consistent high accuracy

Project objective

- To explore the potential of implementing the GoogLeNet and ShuffleNet architectures for fault detection (classification) of an induction motor using thermal imaging, and compare the results with the existing literature

Dataset

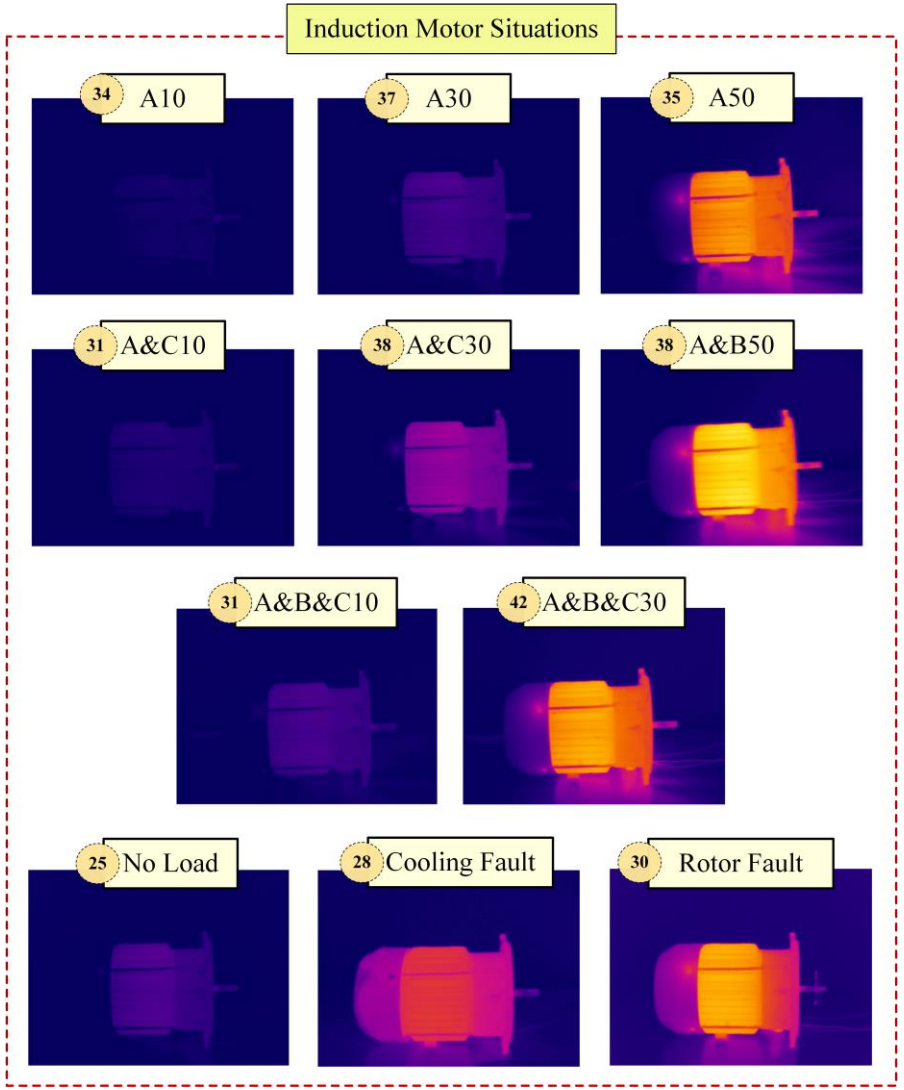
Dataset: Data source

- It comprises thermal infrared images that are used for condition monitoring of induction motors
- The faults are artificially created defects that are internal and do not rely on external parts or failures in the initial setup components
- The acquisition of thermal images was carried out at a workbench using a Dali-tech T4/T8 infrared thermal image camera, in an Electrical Machines Laboratory at the environment temperature of 23°C

Number of phases	3
Connection	Y
Power	1.1 kW
Voltage	380 V
Current	5 A
Speed	2,800 r.p.m.
Frequency	50 Hz

Dataset: Data description

- Consists of **369** images of dimensions 320x240
- It is categorized into **11** classes, each representing a distinct fault condition
- Healthy condition + Two types of failures: Stator (electrical) and rotor (mechanical)



Dataset: Data description

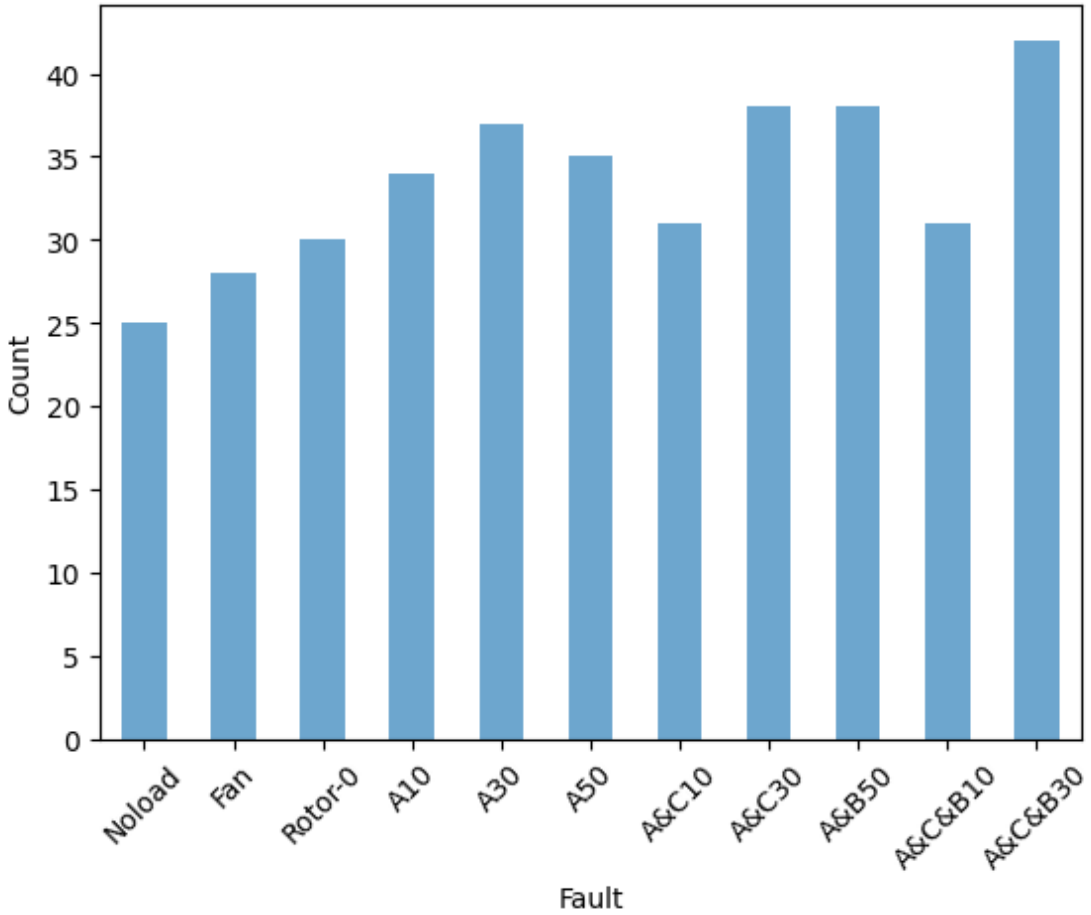


Figure: Distribution of faults in the dataset

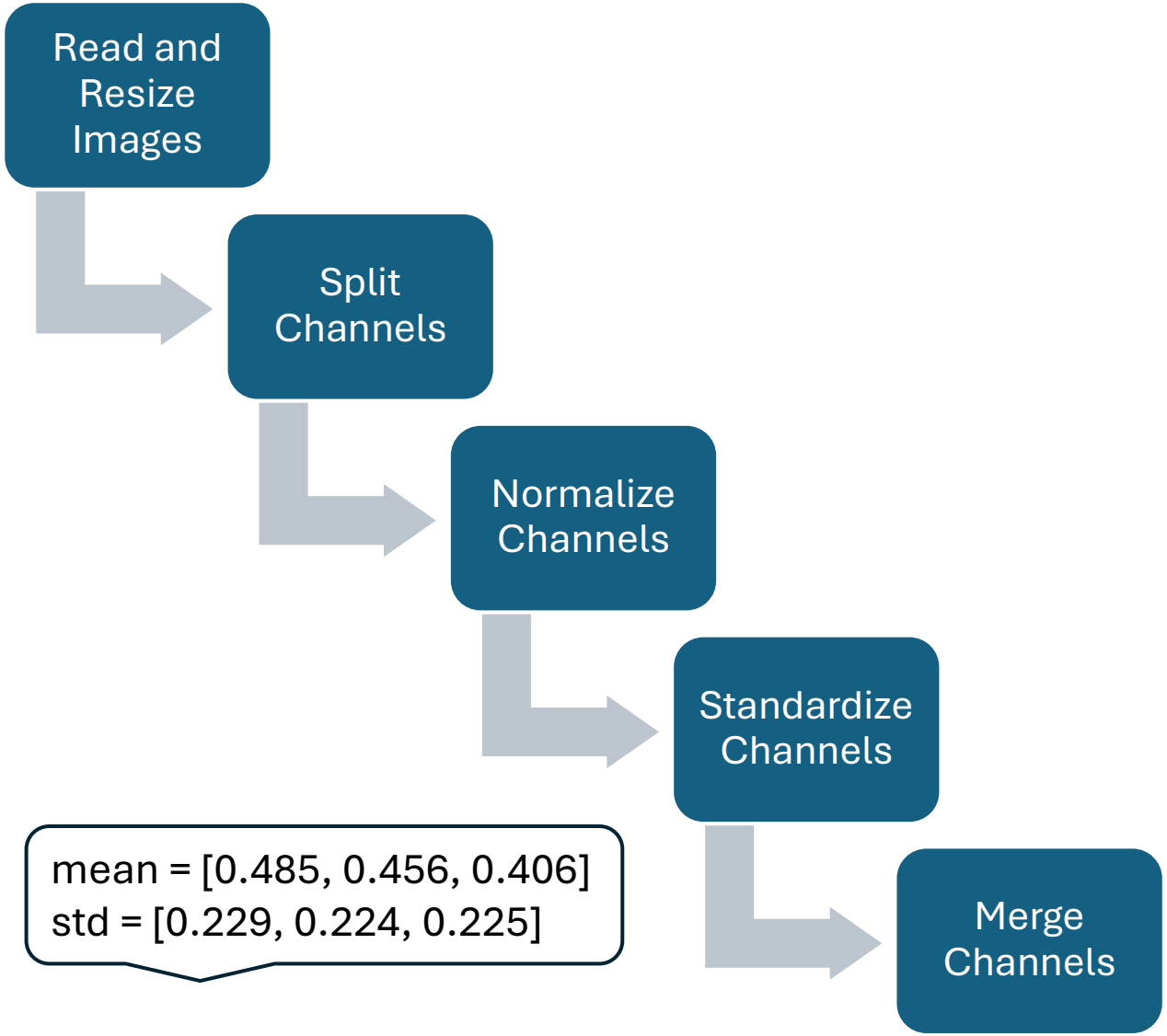
Preprocessing

Preprocessing

- Objective: To ensure images are uniformly scaled and their pixel intensity values are standardized

- Expected input requirements of the pre-trained models:
  - ✓ Mini-batches of 3-channel RGB images
  - ✓ Dimensions: At least 224x224 pixels
  - ✓ Normalization: Range of [0, 1]
  - ✓ Standardization: Using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]

Preprocessing: Main steps




```python
def preprocess_images(data, mean, std, image_size=(320, 240)):
    processed_images = []

    # Read and Resize Images
    for img_path, label in data:
        img = cv2.imread(img_path)
        img = cv2.resize(img, image_size)

        # Split channels
        b, g, r = cv2.split(img)

        # Normalize each channel
        b = b / 255
        g = g / 255
        r = r / 255

        # Standardize each channel
        b = (b - mean[0]) / std[0]
        g = (g - mean[1]) / std[1]
        r = (r - mean[2]) / std[2]

        # Merge channels
        processed_img = cv2.merge([r, g, b])
        processed_images.append([processed_img, label])

    return processed_images
```

Methodology

Methodology: Architectures

Model Architectures

- ✓ Motivation: to implement a simpler architecture, which could offer benefits in terms of training and inference time, for deployment in limited computing power environments (smartphones, tablets)

- ✓ Context: to analyze the impact of these architectures on performance metrics, and compare them with ResNet-50

GoogleNET

SHUFFLENET V2

Methodology: Architectures

GoogleNet
(Inception-v1)

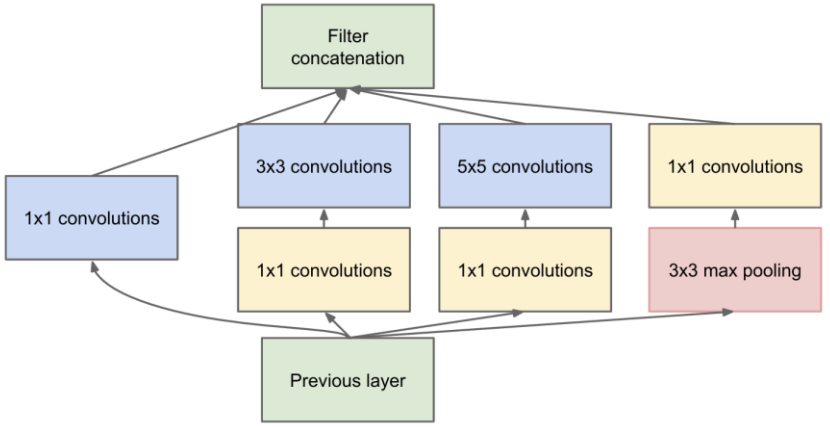
Introduced by: Szegedy et al. (2015)

Key Features:

Inception Modules: Captures features at multiple scales

Efficiency: Suitable for environments with limited computational resources

Parameters: 5,599,904 (excluding fully-connected layer)



(b) Inception module with dimensionality reduction

Methodology

ShuffleNet (ShuffleNetV2)

Introduced by: Ma et al. (2018)

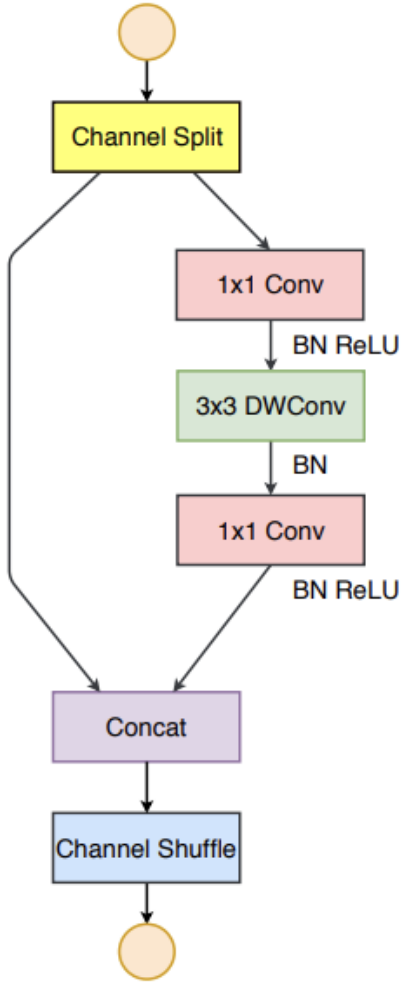
Key Features:

Channel split: Reduces computational complexity

Channel Shuffle: Ensures even distribution of information across feature maps

Suitability for Edge Devices: Ideal for real-time fault detection in constrained environments

Parameters: 1,253,604 (excluding fully-connected layer)



Methodology: Fully Connected Layer

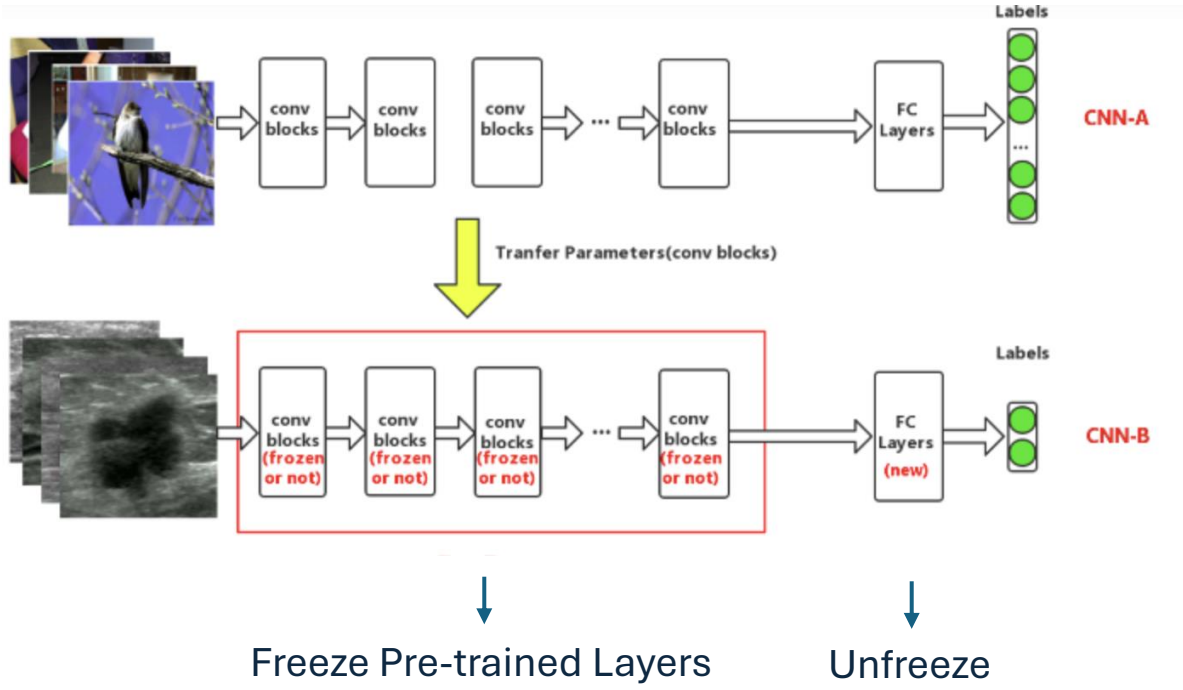
Objective: Replacement of Original Fully Connected Layers

- ✓ Customized layers: 1024-512-128-32-11
- ✓ ReLU activation for hidden layers
- ✓ Softmax for final layer: Probability Distribution over the class labels

	GoogleNet	ShuffleNet
Parameters	5,599,904	1,253,604
Fully Connected	594,955	
Total	**6,194,859**	**1,848,559**

Methodology: Transfer Learning

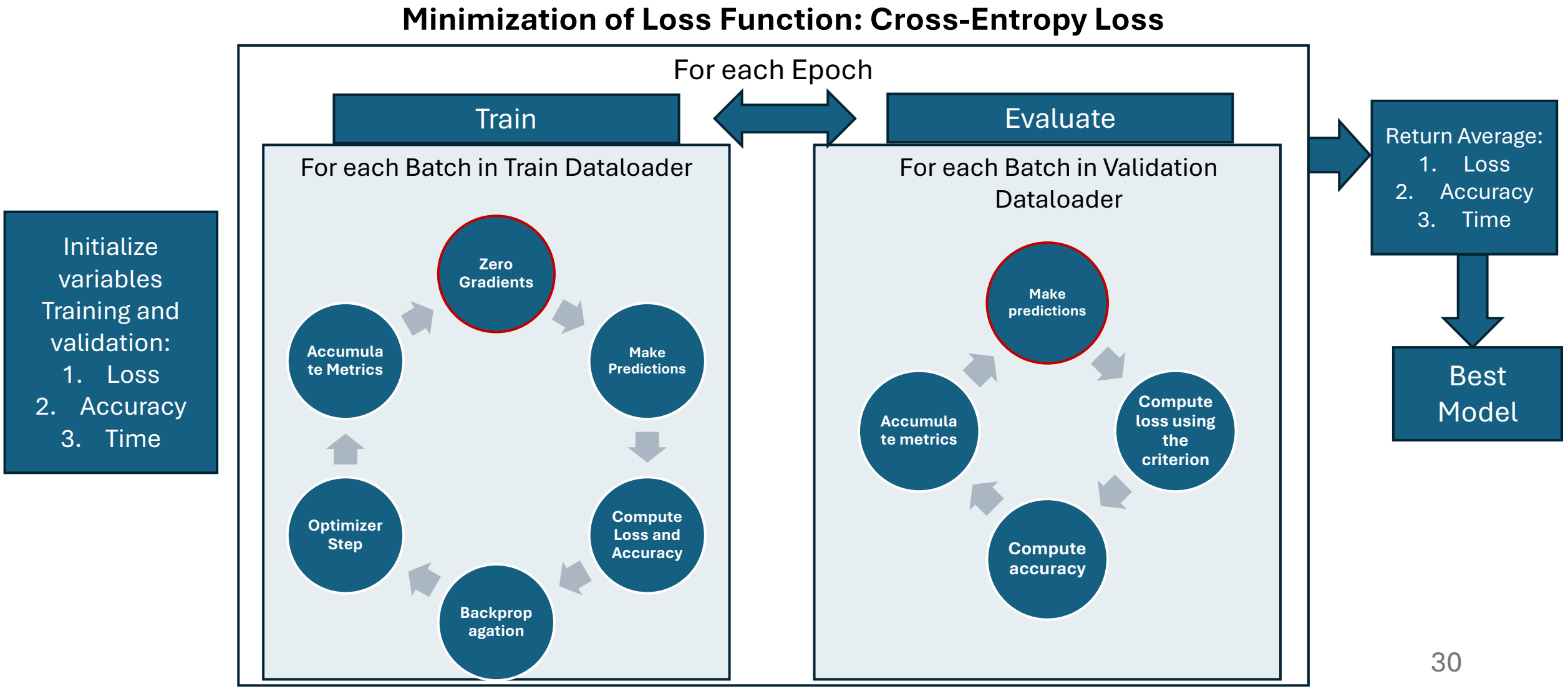
- Leverage Pre-trained Models
  - ✓ Utilize GoogLeNet and ShuffleNet to reduce computational load and training time
- Steps Involved:
  1. Initialize the Pretrained Model
  2. Reshape the Final Layer
  3. Freeze Pre-trained Layers
     - Freeze all layers except the fully connected layer
     - Retain learned features from the pre-trained model
  4. Fine-tune the Last Layer
     - Unfreeze and fine-tune fully connected layer



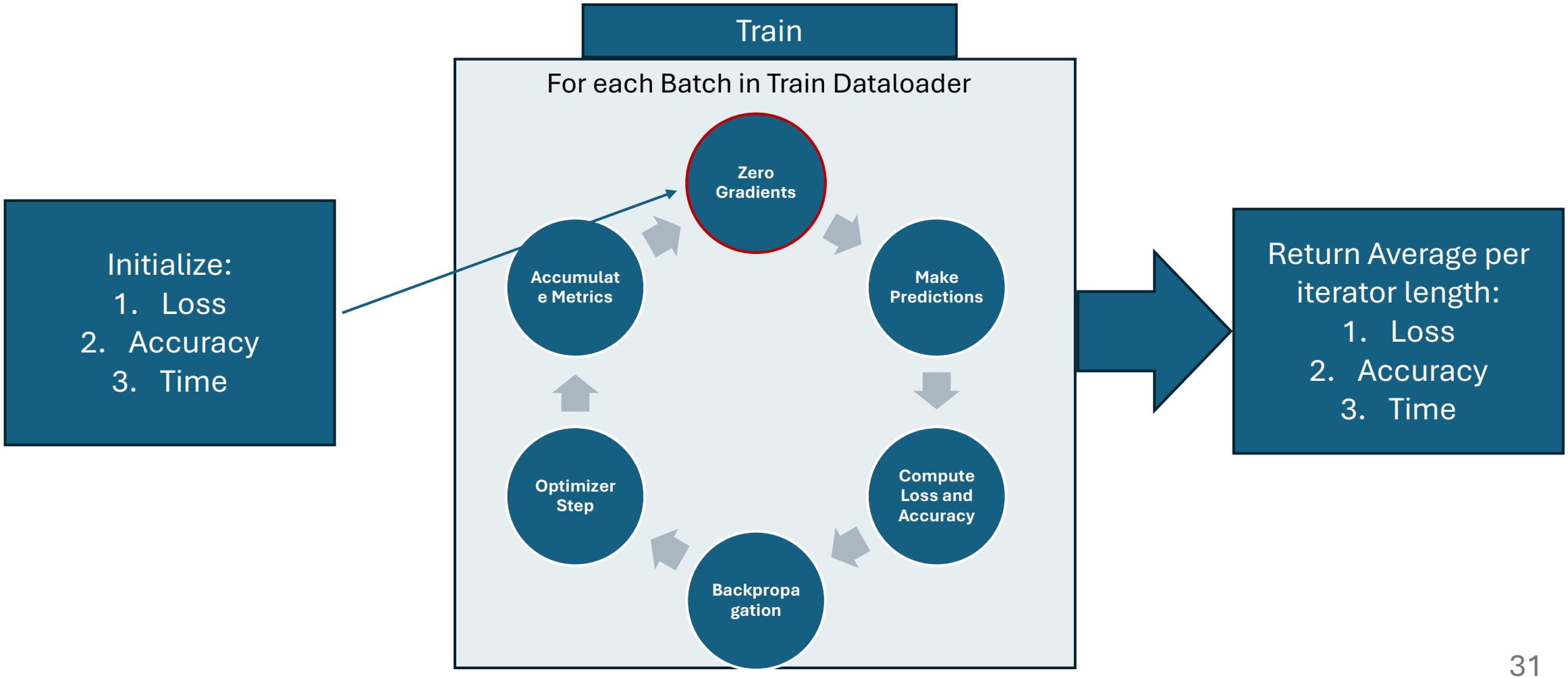
Methodology: Dataset Preparation

1. Download and Extract images
2. Preprocessing
3. Data Splitting:
   - Training and Test Sets: 80-20% ratio
   - Further Splitting: Training set split into training and validation sets (80-20% ratio)
   - Stratified Splitting: Ensures each class is adequately represented
   - Final Counts:
     - Training: **236** images
     - Validation: **59** images
     - Test: **74** images
4. Dataloader: Batching and Shuffling
   - Same batch size for all the sets, shuffle = True only for training

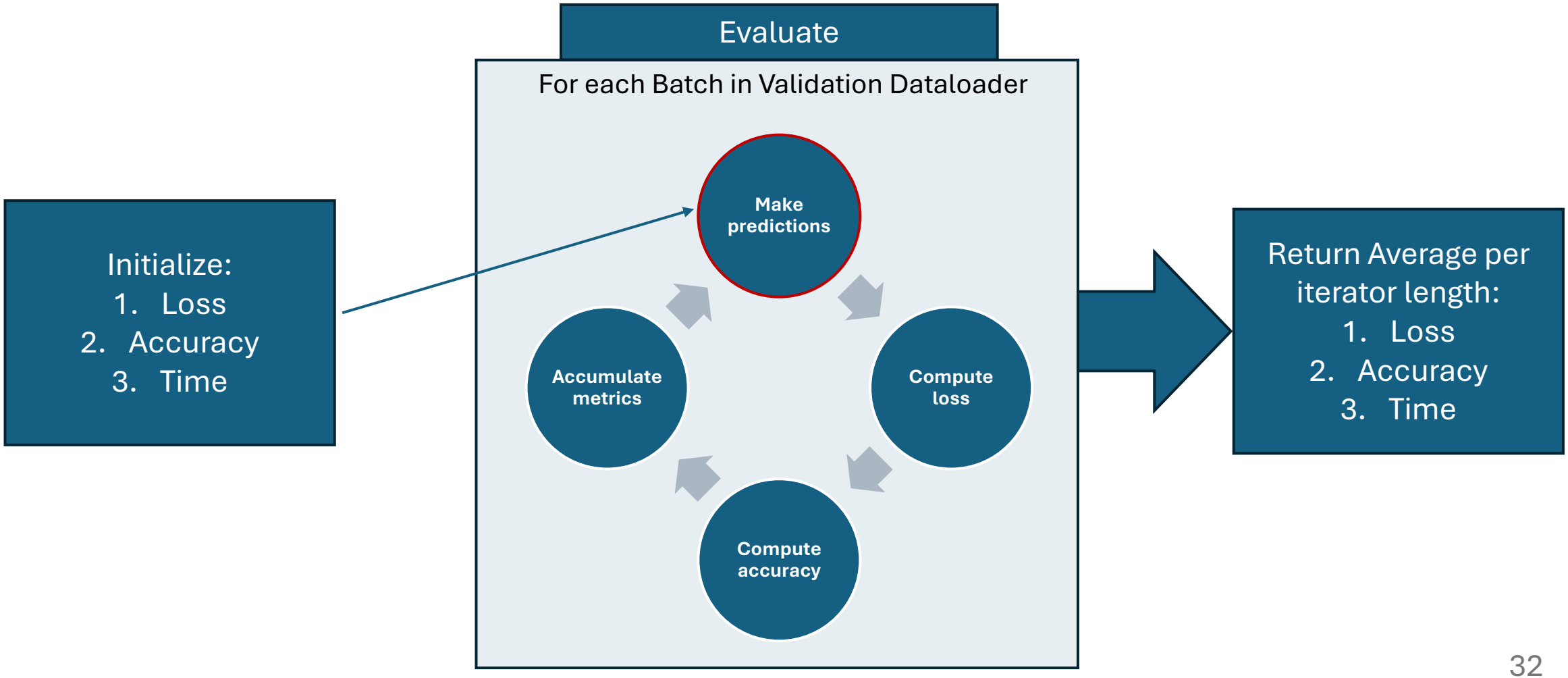
Methodology: Training Procedure



Methodology: Training Procedure



Methodology: Training Procedure



Methodology: Training Procedure

- Performance Metrics:
  - Accuracy
  - Precision
  - Recall
  - F1-Score
  - Confusion Matrix

- Comprehensive Evaluation:
  - Loss and accuracy calculated for training, validation, and test sets.
  - Mean training time per epoch and inference time measured to compare performance among architectures.

Experiments and Results

Experiments: Hyperparameter Tuning

- Combinations

Parameter	Values or Ranges
Learning rate	10^{-3}
Epochs	60
Optimizer	Adam, SGD
Batch Size	8, 16, 32
Dropout	0, 0.2, 0.3

Experiments: Hyperparameter Tuning

- Best hyperparameters

Architecture	Batch size	Optimizer	Epochs best	Dropout
GoogleNet	8	Adam	58	0
ShuffleNet	16	Adam	56	0

Experiments: Results (transfer learning)

- Training set

Architecture	Train accuracy	Train loss	Mean training time per epoch [s]	Validation accuracy	Validation loss
GoogleNet	0.97	0.0805	18.71	1.00	0.0028
ShuffleNet	0.98	0.0711	1.95	0.97	0.0805

- Test set

Architecture	Test accuracy	Test loss	Inference time [s]
GoogleNet	0.97	0.146	4.03
ShuffleNet	0.97	0.0710	0.42

Experiments: Results (transfer learning)

- Confusion matrix

	A30	NoLoad	A50	AC10	AB50	Rotor-0	A10	Fan	AC30	ACB30	ACB10
A30	7	0	0	0	0	0	0	0	0	0	0
NoLoad	0	8	0	0	0	0	0	0	0	0	0
A50	0	0	6	0	0	0	0	0	0	0	0
AC10	0	0	0	7	0	0	0	0	0	0	0
AB50	0	0	0	0	5	0	0	0	0	0	0
Rotor-0	0	0	0	0	0	6	0	0	0	0	0
A10	0	0	0	0	0	0	6	0	0	0	0
Fan	0	0	0	0	0	0	0	8	0	0	0
AC30	0	0	0	0	0	0	0	0	8	0	0
ACB30	0	0	0	0	0	0	0	0	0	5	1
ACB10	0	0	0	0	0	0	0	0	0	1	6

Experiments: Results (transfer learning)

- Training and validation losses: GoogleNet, ShuffleNet

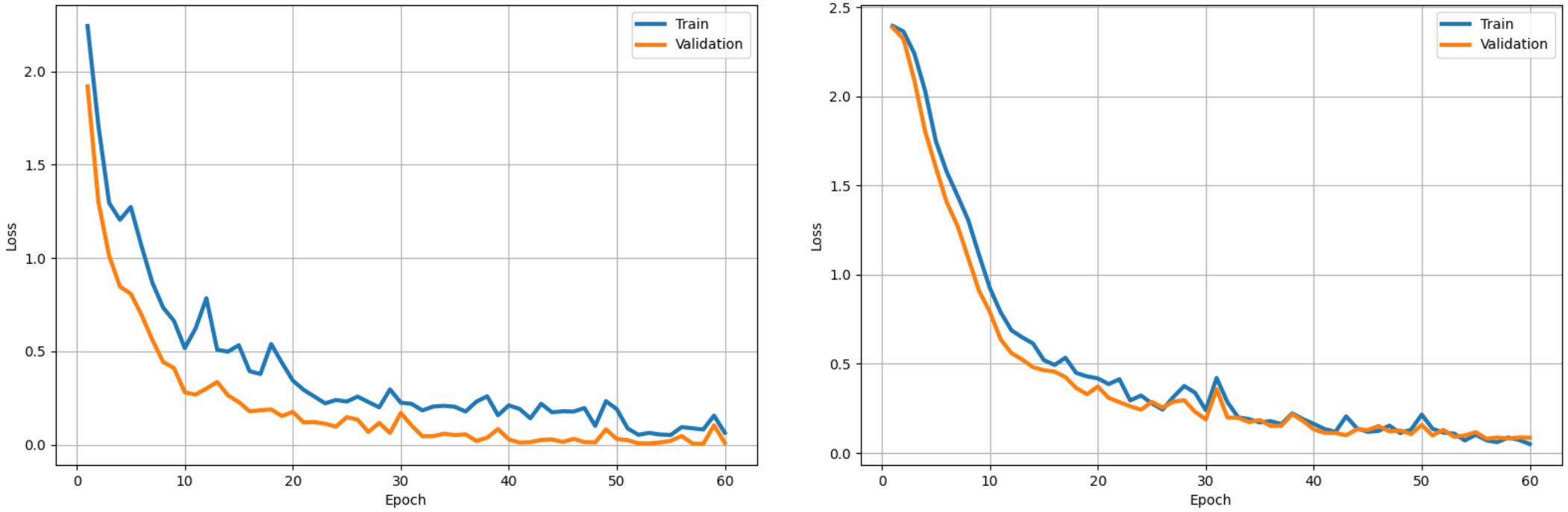


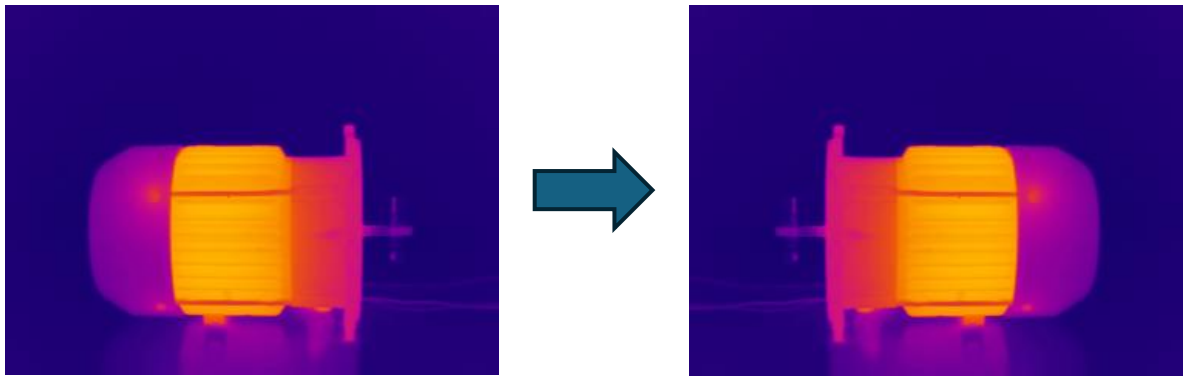
Figure: Training and validation loss for GoogleNet (left), and ShuffleNet (right)

Experiments: Results (transfer learning)

- Two Hypothesis:
  - Validation samples might be less challenging than those used for training → Data augmentation
  - Since transfer learning is employed, the network might have already learned similar features → Train from scratch

Experiments: Data augmentation

- Data augmentation is performed on the training set
- Total number of images is kept constant



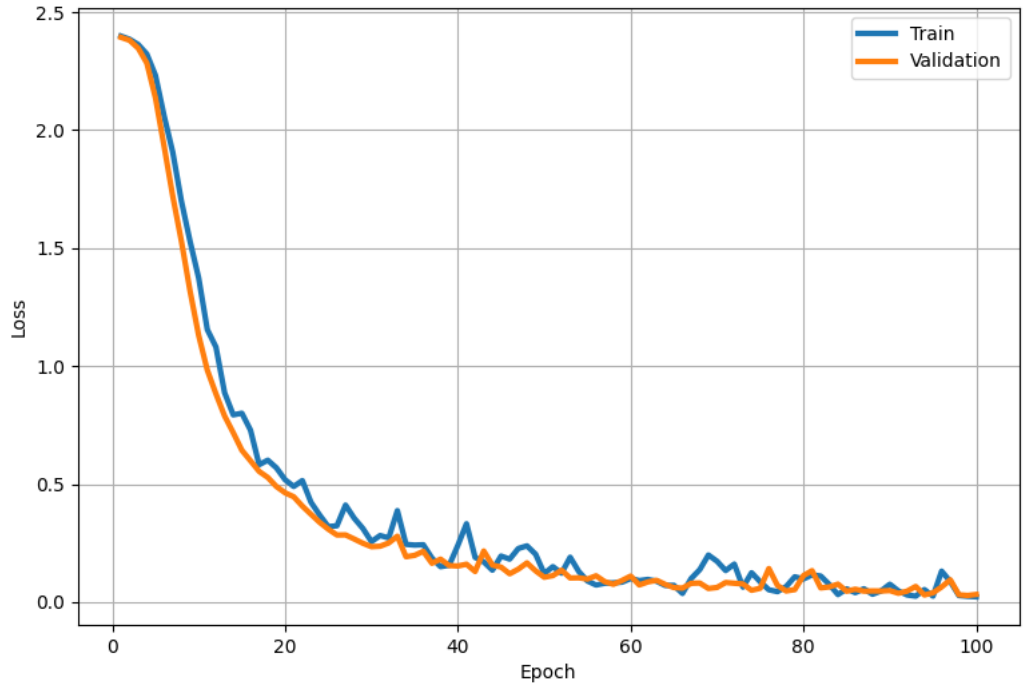


Figure: Training and validation loss for ShuffleNet

Experiments: Training from scratch

- All parameters are unfreezed
- Learning rate set to 10^{-4}
- Previously observed behavior is probably due to the network having already learned similar features from its initial training set
- Test accuracy of 0.82

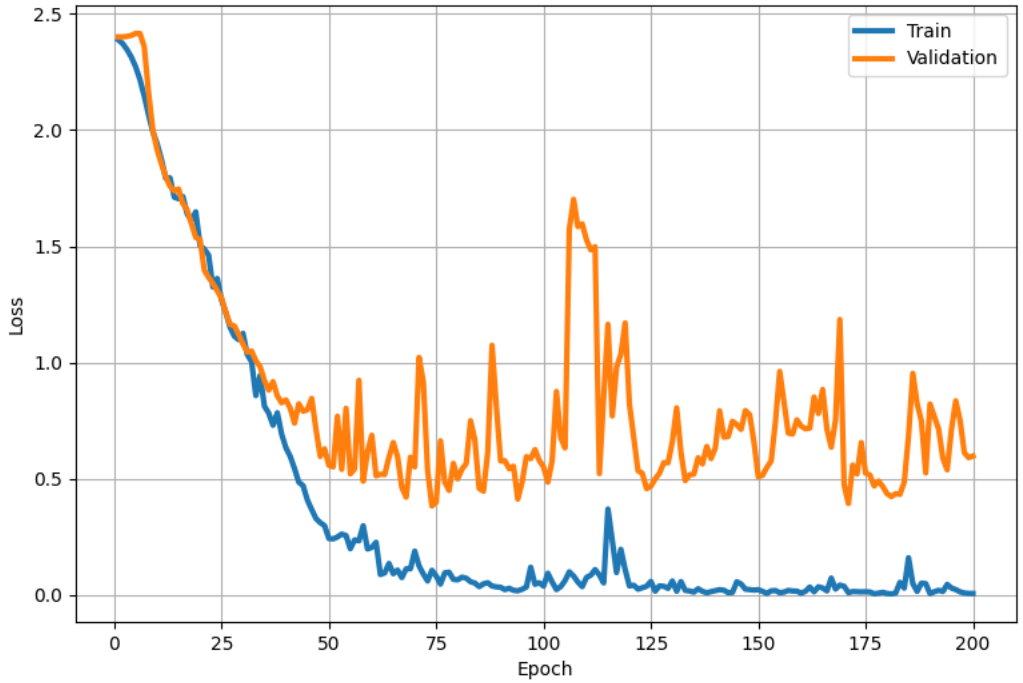


Figure: Training and validation losses for ShuffleNet training from scratch considering data augmenation

Comparison with previous works

Paper	Method Architecture	Accuracy
Najafi et al. (2020)	Decision Trees	0.938
Sakalli et al. (2020)	ResNet50	1.00
This study	GoogleNet	0.97
	ShuffleNet	0.97
	ShuffleNet (from scratch)	0.82

Conclusions and Future Work

Conclusions

- The study focused on using GoogleNet and ShuffleNet architectures with transfer learning to diagnose faults in asynchronous motors through thermal imaging
- Both models achieved a high accuracy of 97% on the test set, comparable to the more complex ResNet50, while ShuffleNet offered faster training and inference times due to its lightweight design
- Transfer learning significantly enhanced performance, with ShuffleNet trained from scratch achieving only 82% accuracy

Future Work

- Advanced data augmentation techniques
- Real-time implementation in industrial settings
- Expanding the dataset to include other equipment like transformers or higher power motors

Fault Diagnosis of Asynchronous Motor through Thermal Imaging using Convolutional Neural Networks

Vision and Cognitive Systems Project

June 25th 2024

Alejandra Cruces
Mario Tapia

CrossEntropy Loss Function

$$\ell(x, y) = L = \begin{bmatrix} l_1 \\ \vdots \\ l_N \end{bmatrix}^T$$

$$l_n = -\log\left(\frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C}\exp(x_n, c)}\right) \cdot 1_{\{y_n \neq \text{ignore\_index}\}}$$

- x: Input (e.g., logits from a neural network)
- y: Target class index
- y_n: Target class index for the n-th sample
- N: Number of samples (batch size)
- C: Number of classes
- $\exp(x_{n,y_n})$: Exponential of the logit for class y_n
- $\sum_{c=1}^{C}\exp(x_n, c)$: Sum of exponentials over all classes
- $1_{\{y_n \neq \text{ignore\_index}\}}$: Indicator function (1 if y_n is not equal to ignore_index, 0 otherwise)