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

Alejandra Cruces
alejandraolivia.crucesandrews@studenti.unipd.it

Mario Tapia
marioalejandro.tapiamontero@studenti.unipd.it

Abstract

This project classifies thermographic images of induction motors failures using GoogleNet and ShuffleNet convolutional neural networks with transfer learning. Both models achieved high test set accuracy of 97%, with ShuffleNet offering faster training and inference times, making it ideal for resource-limited environments. Transfer learning significantly enhanced performance over training from scratch. The results were competitive with previous studies using more complex models. This study highlights the potential of CNNs with transfer learning for efficient and accurate industrial fault diagnosis.

1. Introduction

Electric motors have broad applications in such areas as industry, business, public service, and household electrical appliances. They power a variety of equipment including wind blowers, water pumps, compressors, and machine tools. In industrially developed nations and large developing nations, electric motors account for a considerable proportion of total national power consumption. Statistics indicate that electric motors are generally responsible for about 2/3 of industrial power consumption in each nation, or about 40% of overall power consumption [4]. Among these machines, the three-phase induction motor is the most commonly used type for providing motive power due to its reliability and low cost [7, 8]. Some of the advantages of early fault detection in induction motors are as follows [2]:

- Cost savings which are realized by estimating potential failures before they occur
- Facilitate pre-planned preventive machine schedules
- Prevent unexpected stop in the production lines
- Improve the induction motor efficiency

Electrical machines and drive systems are subject to many different types of faults. These faults include the following:

- Stator faults which are defined by stator winding open or short circuited
- Rotor electrical faults which include rotor winding open or short circuited for wound rotor machines and 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

This work focuses on stator faults (short circuit), and rotor mechanical fault (locked-rotor condition and cooling failure).

When induction motors are running, each type of fault generates characteristic features in the machine's behavior. Fault detection techniques can be based on the analysis of vibrations, stator current, shaft speed, acoustic emissions, voltage, and temperature (infrared thermography), among others. An ideal diagnostic procedure should take the minimum measurements necessary from a machine and by analysis extract a diagnosis, so that its condition can be inferred to give a clear indication of incipient failure modes in a minimum time [1]. Based on the latter, this project aims 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. The motivation behind this investigation is to implement a simpler architecture, which could offer benefits in terms of training and inference time, and to analyze the impact of these architectures on performance metrics such as accuracy.

2. Related Work

The project focuses on the essential task of non-destructive fault detection and prediction, crucial for effective industrial equipment condition monitoring. To accomplish this, the project leverages the Infrared thermal imaging of Induction motor dataset introduced by [3]. In their study, utilized a multi-step model: segmentation using Random Forest and AdaBoost, feature extraction based on first- and second-order statistics, and classification through a Decision Tree classifier. Their methodology addressed 11 different fault classes, achieving an accuracy of 93.8% using a 50%-50% training-test split, with training further validated by a 10-fold cross-validation approach.

Using the same dataset, [5] adopt a distinct approach, utilizing transfer learning based on a deep learning architecture: ResNet50. In their experiments, employ two test methods (70%-30% and 80%-20% training-test splits) with varying hyperparameters. The best results are obtained using the SGD optimizer with high learning rates and the Adam optimizer with a low learning rate of 0.0001. The 80%-20% split generally yields better scores, with a low learning rate providing the most consistent high accuracy.

Both papers significantly contribute to the advancement of fault diagnosis in electrical equipment through thermal imaging analysis, with non-invasive methodologies. While [3] offers valuable insights into dataset creation and interpretable machine learning techniques, [5] highlights the efficacy of transfer learning with deep learning models in fault identification.

3. Dataset

3.1. Data Source

As it was mentioned, the dataset used in this study was first presented in [3]. 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.

The electrical machine under fault corresponds to a three-phase induction motor whose specifications are presented in Table 1.

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

Table 1. Induction motor specifications.

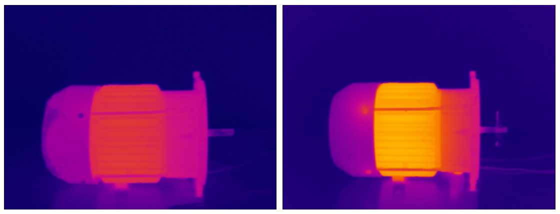
Figure 1. Motor's heat distribution under fan failure (left), and locked-rotor condition (right).

3.2. Data Description

The dataset consists of 369 images of dimensions 320x240 in bmp format. It is categorized into 11 classes, each representing a distinct fault condition. These include:

- No-load, which indicates a healthy state
- Fan, which signifies a cooling failure
- Rotor-0, which represents a locked-rotor condition
- Various short circuit conditions, which are denoted by a combination of letters and numbers. In this notation, the letters indicate the phases where the short circuit occurred, and the number represents the current level (as a percentage of the short-circuit current)

As an example, Figure 1 shows the heat distribution of the motor under fan and locked-rotor failures.

The distribution of classes within the dataset is depicted in Figure 2. This visualization confirms that the dataset is balanced, with no class being over- or under-represented.

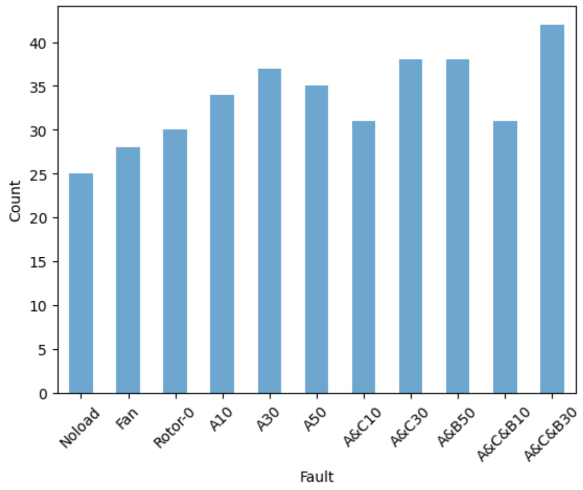

Figure 2. Distribution of faults in the dataset.

3.3. Preprocessing

For the preprocessing step, standard procedures are followed to ensure images are uniformly scaled and their pixel intensity values are standardized. Additionally, the expected input requirements of the pre-trained models (GoogLeNet and ShuffleNet) are considered to make the images suitable for model training. These models require mini-batches of 3-channel RGB images with dimensions of at least 224x224 pixels. The images need to be loaded in the range of [0, 1] and then standardized using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]. Taking this into account, the main steps are:

1. Read and Resize Images: Each image is read from its file path and resized to 320x240 pixels.
2. Split Channels: The image is split into its three color channels: blue (B), green (G), and red (R).
3. Normalize Channels: Each channel is normalized by dividing the pixel values by 255, converting the range from 0-255 to 0-1.
4. Standardize Channels: Each normalized channel is standardized using the provided mean and standard deviation values (mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]). The standardization formula is:

$$\text{standardized\_channel} = \frac{\text{normalized\_channel} - \text{mean}}{\text{std}}$$

5. Merge Channels: The standardized channels are merged back together in the order of red, green, blue (RGB).

4. Method

The following section presents and explains the methodology used to carry out this work.

4.1. Model Architecture

In this study, we utilized two architectures: GoogleNet and ShuffleNet. These models were chosen because we wanted to investigate the performance in fault detection using smaller models compared to [5], were a ResNet-50 architecture was used. The primary motivation for this is to enable focus on tasks like fault diagnosis at worksites using common devices such as smartphones, where computing power is limited and rapid inference time is crucial.

4.1.1 GoogleNet

GoogleNet is a convolutional neural network architecture introduced by [6] in 2015. For this work, Inception-v1 is used. Important features of this network are the following:

- **Inception Modules:** GoogleNet uses inception modules to capture features at multiple scales, which is particularly useful for analyzing thermographic images with varying fault sizes and patterns.
- **Efficiency:** The architecture's efficient use of parameters makes it suitable for deployment in environments with limited computational resources, such as industrial settings.
- **Performance:** GoogleNet has demonstrated strong performance on various image classification benchmarks, which supports its use in our fault classification task.
- **Parameters:** Without the fully-connected layer, the number of parameters corresponds to 5,599,904.

4.1.2 ShuffleNet

ShuffleNet is a lightweight CNN architecture designed for mobile and embedded applications. Introduced by [9] in 2018. For this work, ShuffleNetV2 is used. Important features of this network are the following:

- **Pointwise Group Convolution:** This technique reduces computational complexity by dividing the convolution into smaller groups, making it more efficient while maintaining accuracy.
- **Channel Shuffle:** The channel shuffle operation ensures that information is evenly distributed across the feature maps, enhancing the model's representational capacity.
- **Suitability for Edge Devices:** Due to its lightweight nature, ShuffleNet is well-suited for real-time fault detection in industrial edge devices, where computational resources may be constrained.
- **Parameters:** Without the fully-connected layer, the number of parameters corresponds to 1,253,604.

4.1.3 Modification of Fully Connected Layer

To adapt GoogleNet and ShuffleNet for our specific classification task, we replaced their original fully connected layers to match the number of output classes. The sequence of fully connected layers is configured as 1024-512-128-32-11. Each layer employs the ReLU as activation function, and a softmax activation for the final layer to provide a probability distribution over the class labels. The total number of parameters, including the fully connected layers, is 6,194,859 for GoogleNet and 1,848,559 for ShuffleNet.

4.1.4 Transfer Learning Approach

To leverage the pre-trained capabilities of GoogLeNet and ShuffleNet and reduce the computational load and training time, feature extraction was employed, modifying and updating only the fully connected layers. This approach involves the following steps:

- **Initialize the pretrained model:** The pre-trained model is loaded.
- **Reshape the final layer:** The original fully connected layer is replaced with a customized fully connected layer to match the number of output classes. This new layer has progressively decreasing neuron counts, alternating with ReLU activation functions and dropout layers to prevent overfitting..
- **Freeze pre-trained layers:** Initially, all layers in the network except the fully connected layer are frozen. This prevents the weights of these layers from being updated during training, allowing retention of the learned features from the pre-trained model.
- **Fine-tune the last layer:** The parameters of the fully connected layer are then unfrozen to fine-tune it for the specific classification task. This step adapts the model to output predictions that match the number of classes in the dataset.

4.2. Hyperparameter Tuning

To identify the optimal hyperparameters for our models, we conducted a series of experiments varying batch size, optimizer, and dropout rate. The combinations of hyperparameters showed in Table 2 were tested:

Architecture	Epochs	Optimizer	Batch Size	Dropout
GoogleNet ShuffleNet	60	Adam, SGD	8, 16, 32	0, 0.2, 0.3

Table 2. Hyperparameter combinations tested.

For all experiments, a learning rate of 10^{-3} was used. When using the SGD optimizer, the momentum was set to 0.99.

4.3. Training Procedure

The process begins with the preparation of the dataset, which involves downloading and extracting images. These images then undergo preprocessing, as previously detailed. Following this, the data is divided into training and test sets. To ensure that each class is adequately represented, we employ stratified splitting. The dataset is partitioned into training and test sets in an 80-20% ratio, respectively. The training set is further subdivided into training and validation sets, also using the same split proportion. This leads to a training, validation, and test sets with 236, 59, and 74 images, respectively.

During training, the Cross-Entropy Loss function is minimized using several combination of hyperparameters from 2. The training loop iterates over the dataset for a set number of epochs, where `optimizer.zero_grad()` is called at the start of each iteration to reset the gradients.

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

Table 3. Best hyperparameters combination for each architecture.

The forward pass computes the model's predictions, and the loss is calculated with `criterion(output, target)` using the Cross-Entropy Loss function. `loss.backward()` is then used to compute the gradient of the loss with respect to the model parameters, and `optimizer.step()` is called to update the model's parameters based on these gradients.

For inference, `torch.no_grad()` is used to disable gradient computation, reducing memory usage and improving inference speed. The model's performance is evaluated using accuracy, precision, recall, F1-score, and confusion matrix. The `calculate_accuracy` function is employed to compute the accuracy of the model by comparing the predicted labels with the actual labels. The loss and accuracy are calculated for the training, validation, and test sets to ensure comprehensive evaluation. Additionally, the mean training time per epoch and inference time are measured to compare performance among different architectures. This comprehensive approach ensures the model is well-tuned and performs effectively on unseen data.

5. Experiments

The following section presents and analyzes the main experiments and results obtained.

5.1. Transfer Learning

5.1.1 Results

The best models found after hyperparameter tuning are displayed in Table 3. The results for the training set of each model are shown in Table 4, and the test set results are presented in Table 5.

In both architectures, the Adam optimizer yielded the best results, likely due to its adaptive learning rate. Additionally, a dropout rate of 0 produced the best results, suggesting that overfitting is not an issue. This indicates that dropout may be removing important features captured by the network, thereby worsening generalization.

It can be observed that both architectures perform well, achieving very high validation set accuracies, above 0.97. This ability to generalize to new instances is further confirmed by the test accuracies, achieving 0.97 for both architectures.

It is important to highlight that the smaller number of parameters in ShuffleNet is reflected in the mean training

Architecture	Train accuracy	Train loss	Mean training time per epoch [s]	Validation accuracy	Validation loss	Validation F1-score
GoogleNet	0.97	0.0805	18.71	1.00	0.0028	1.00
ShuffleNet	0.98	0.0711	1.95	0.97	0.0805	0.97

Table 4. Results for the training set using transfer learning.

Architecture	Test accuracy	Test loss	Inference time [s]	Test precision	Test recall	Test F1-score
GoogleNet	0.97	0.146	4.03	0.97	0.97	0.97
ShuffleNet	0.97	0.0710	0.42	0.97	0.97	0.97

Table 5. Results for the test set using transfer learning.

	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

Table 6. Confusion matrix

time per epoch and inference times, where it performs several times faster than GoogleNet. This behavior, though influenced by differences in batch sizes, is primarily due to differences in the complexity of the architectures.

Table 6 shows the confusion matrix for both models, indicating that there is one misclassified example for each one of the three-phase short circuits, which only differ in the intensity of the short circuit failure.

5.1.2 On the training and validation losses result

Figure 3 displays the losses for both the training and validation sets across the two architectures. Interestingly, the validation loss is lower than the training loss in both instances. Normally, we expect the training loss to be lower than the validation loss during training because the model is not exposed to the validation data. Our hypothesis suggests two possible explanations for the observed phenomenon.

Firstly, the validation samples might be less challenging than those used for training. This is plausible since the images are taken to the same machine in the same position, typical in industrial or experimental setups. To test this, data augmentation is applied to the training set to increase its difficulty and variability.

Secondly, since transfer learning is employed, the network might have already learned similar features from its pre-training data, making it easier to predict the correct failure class. To investigate this, the model is trained from scratch by unfreezing the CNN parameters during training and increasing the number of epochs (due to limitations in time and resources, the experiments are performed exclusively using the ShuffleNet architecture).

5.2. Data augmentation

Data augmentation on the training set is implemented by randomly flipping the examples horizontally. The validation and test sets, as well as the total number of samples, remain unchanged. Figure 4 (top) illustrates the loss curves, which exhibit the same behavior observed in the bottom plot of Figure 3.

5.3. Training from scratch

In the subsequent experiment, the network was trained from scratch, incorporating data augmentation and starting with an untrained model. The learning rate was set to 10^{-4}, and the number of epochs was adjusted accordingly. This adjustment was made because the validation loss curve exhibited numerous spikes, likely caused by an initially high learning rate. The results, shown in Figure 4 (bottom), confirm that the previously observed behavior was probably due to the network having already learned similar features from its initial training set, leading to good performance on the validation set. In this case, the best combination of hyperparameters led to a test accuracy of 0.82.

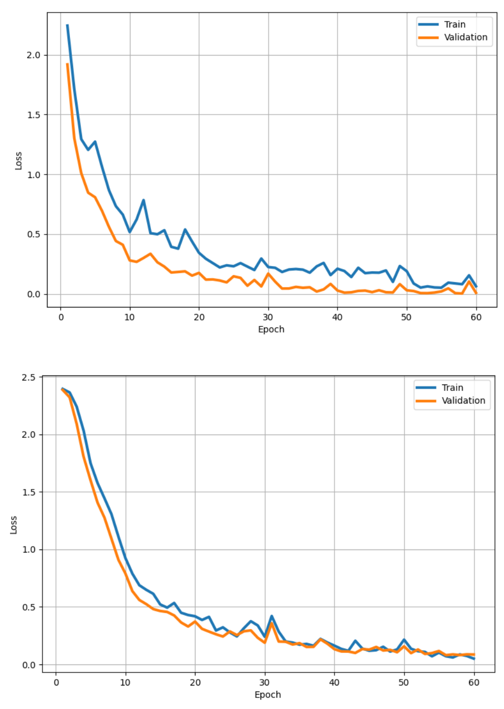

Figure 3. Training and validation loss for GoogleNet (top), and ShuffleNet (bottom)

Paper	Method/Architecture	Accuracy
[3]	Decision trees	0.938
[5]	ResNet50	1.00
This study	GoogleNet	0.97
This study	ShuffleNet	0.97
This study	ShuffleNet (from scratch)	0.82

Table 7. Comparison between our results and previous works.

5.4. Comparison with previous works

Table 7 presents a comparison of our results with those mentioned in the Related Work section who used the same dataset. Similar to [5], we employed a 80%-20% training-test split, where ResNet50 achieved higher accuracy. However, ShuffleNet, optimized for efficient computation and low memory usage, offers slightly lower accuracy but is more suitable for real-world industrial applications. This trade-off ensures practical feasibility without compromising substantial performance gains. In contrast, [3] utilized decision trees, known for their interpretability but resulting in lower accuracy compared to deep learning approaches. Our approach, which utilized simpler architectures and transfer learning, demonstrated good performance relative to these previous studies.

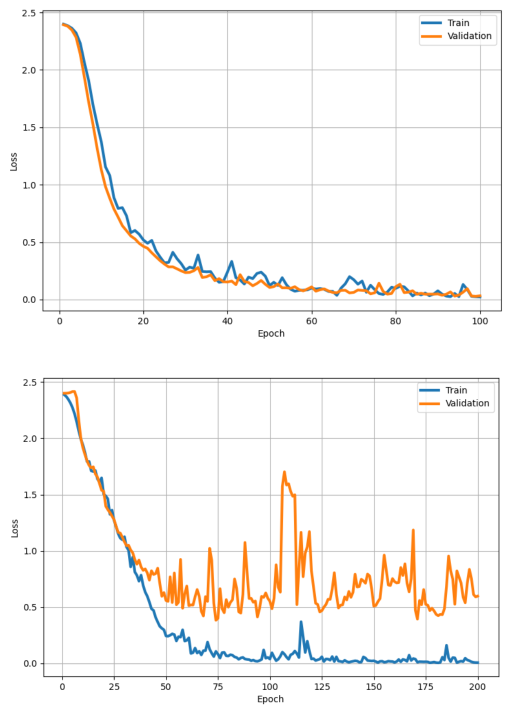

Figure 4. Training and validation losses for ShuffleNet pretrained (top), and trained from scratch (bottom), considering data augmentation.

6. Conclusion

This 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 could explore additional lightweight architectures, advanced data augmentation techniques, real-time implementation in industrial settings, and expanding the dataset to include other equipment like transformers or higher power motors.

References

[1] Alberto Bellini, Fiorenzo Filippetti, Carla Tassoni, and Gérard-André Capolino. Advances in diagnostic techniques for induction machines. *IEEE Transactions on Industrial Electronics*, 55(12):4109–4126, 2008.

[2] Tomas Garcia-Calva, Daniel Morinigo-Sotelo, Vanessa Fernandez-Cavero, and Rene Romero-Troncoso. Early detection of faults in induction motors—a review. *Energies*, 15(21):7855, 2022.

[3] Mohamad Najafi, Yasser Baleghi, Sayyed Asghar Gholamian, and Seyyed Mehdi Mirimani. Fault diagnosis of electrical

equipment through thermal imaging and interpretable machine learning applied on a newly-introduced dataset. In *2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, pages 1–7. IEEE, 2020.

[4] R. Saidur. A review on electrical motors energy use and energy savings. *Renewable and Sustainable Energy Reviews*, 14(3):877–898, 2010.

[5] Gönül Sakalli and Hasan Koyuncu. Categorization of asynchronous motor situations in infrared images: Analyses with resnet50. In *2022 International Conference on Data Analytics for Business and Industry (ICDABI)*, pages 114–118, 2022.

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[7] Oregon State University. Common industrial motor types, n.d. Accessed: 2024-05-24.

[8] Carlos Verucchi, Cristian Ruschetti, and Fernando Benger. Efficiency measurements in induction motors: Comparison of standards. *IEEE Latin America Transactions*, 13(8):2602–2607, 2015.

[9] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.