Towards Automated Resistor Classification: A Machine Learning-Based Approach to Resistor Identification

Philipp Bauerfeind
Faculty of Electrical Engineering and Information Technology
Ostbayerische Technische Hochschule Regensburg
Regensburg, Germany
philipp.bauerfeind@st.oth-regensburg.de

***Abstract*—This paper presents a machine learning approach to the image-based classification of 4- and 5-band resistors, leveraging deep neural networks and transfer learning. To address the challenges posed by a small dataset, a preprocessing pipeline is proposed incorporating a custom-trained YOLOv11 segmentation model, a deep neural network for background removal, and traditional computer vision techniques. By applying transfer learning on pretrained ImageNet models and averaging predictions from flipped input images, the presented method achieves a classification accuracy of 86.61 %.**

***Key Words*—computer vision, machine learning, transfer learning, deep learning, image classification, image segmentation**

I. Introduction

Resistors are among the most common components in electrical circuits. While industrial printed circuit boards (PCBs) predominantly use surface-mounted device (SMD) resistors, hobbyists and makers often use through-hole technology (THT) resistors. Unlike SMD resistors, THT resistors rely on colour bands printed on their bodies for identification [1].

These colour bands provide essential information, including resistance value and tolerance. The sequence of these bands is critical: the first band is typically closest to the resistor's wire leads, and metallic-coloured bands never appear first. [1]

While the colour code system is systematic, it can be time-consuming and prone to errors, especially for inexperienced individuals who may not be able to memorise the codes and must repeatedly refer to look-up tables. To address this challenge, this paper proposes a machine learning-based approach to classify resistors using images. By leveraging inductive transfer learning [2], the trained model can automate resistor identification and has the potential to be integrated into mobile applications, thereby simplifying the process for hobbyists and professionals alike.

II. Related Work

The classification of resistors has been previously addressed, with a predominant focus on the detection of the coloured bands and subsequent resistance calculation.

In 2015, Niklaus and Ulli [3], students of the ETH Zürich, developed an Android App to identify THT resistors as part of a group thesis. The proposed algorithm first identifies the resistor, then detects the colour bands, which is divided into two separate steps, and then classifies the colour bands by comparing the colour of a band to a reference colour. While the initial three steps of the proposed algorithm demonstrated robust functionality, the accuracy of the colour detection method was found to be only 25 %, with a notable deficiency in the identification of brown, orange, yellow, and violet colours.

In 2019, Muminovic et al. [4] also addressed this issue, employing predominantly conventional computer vision methods. They proposed a technique for segmenting the resistor body and subsequently classifying the resistor by extracting information directly from the colour band. For the classification of the colour bands, they utilised a support vector machine (SVM). Their proposed method attained an accuracy of approximately 86 %, while maintaining real-time capabilities. The occurrence of misclassifications of the resistors was predominantly attributed to the detection of colour bands being prone to errors. The results of this study were gathered using exclusively pictures with a white background.

A review of the literature on this topic reveals a significant presence of studies that employ traditional computer vision techniques. While these conventional methods appear to yield encouraging results, the present study focuses on machine-learning algorithms, with a particular emphasis on Convolutional Neural Networks (CNNs). This approach aims to reduce the necessity for preprocessing steps and to train an algorithm capable of classifying images of resistors.

III. Preprocessing Pipeline

The primary dataset used for this project was obtained from Kaggle containing 4- and 5-band resistors [5]. This dataset consists of 127 distinct classes and therefore resistances. The images were captured using an USB microscope, and incorporate five different backgrounds with two distinct leg orientations of the resistor (straight and bent), resulting in 10 images per class.

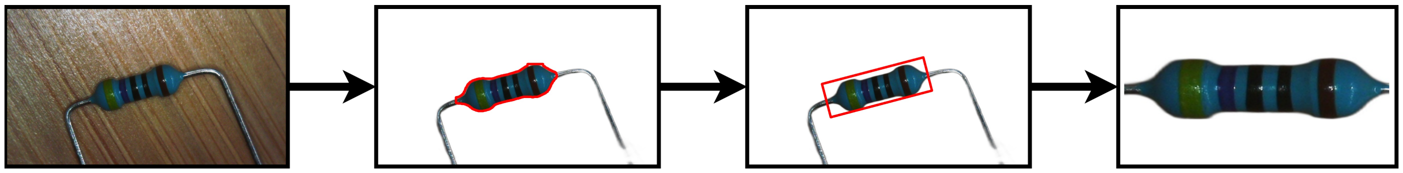

Figure 1: Visualization of the proposed Preprocessing Pipeline (1. Background Removal, 2. Body Segmentation, 3. Detection of minimal Bounding Box and 4. Rotation + Resizing)

Initial experimentation with training CNNs on the existing dataset revealed no success. Consequently, a preprocessing pipeline was proposed that extracts only the body of the resistor, thereby identifying the essential features in the image.

The initial step of the process involves the removal of the background of the image (see Figure 1), leaving only the resistor. The library rembg [6] was used for this purpose, as it offers an AI-powered tool to remove backgrounds.

Consequently, a custom YOLOv11-nano segmentation model [7] is employed to detect the body of the resistor. This model originates from a pretrained YOLO model that was finetuned on 104 training samples. These training samples are drawn from the original Kaggle dataset; however, the background of the images has been removed, and they have been manually labelled using polygon annotations. In addition to the training samples, 30 validation and 15 test samples were created for the purpose of evaluation.

Following the detection of the resistor body, the subsequent step is to identify the bounding box that encloses the body with minimal surface area.

Finally, the bounding box is rotated and resized. It is important to note that the algorithm is unable to determine whether the resistor has been correctly rotated, meaning that the first band of the resistor may not be placed on the left side of the image. The entire process of preprocessing an image takes approximately 4 seconds.

The proposed preprocessing pipeline was applied to derive a new dataset. It should be noted that, on occasion, parts of the resistor may be mistakenly removed, or the detection of the resistor body may result in defective results. Consequently, 91 images were removed from the processed dataset, thereby reducing the dataset to 92.83 % of its original size. Of this dataset, 925 images were utilised for training purposes, while 127 images, one for each class, were allocated for validation and testing alike.

IV. Classification

Following the construction of a preprocessing pipeline and the generation of a derived dataset, the objective is to train a CNN. Due to the limited availability of data, transfer learning is essential. Fortunately, Keras provides a diverse range of models pretrained on the ImageNet dataset [8].

A. General Training Procedure

In order to compare the performance of these base models in our target domain, a similar training strategy is employed. The model is fed with a 256x256 RGB image and a batch size of 20

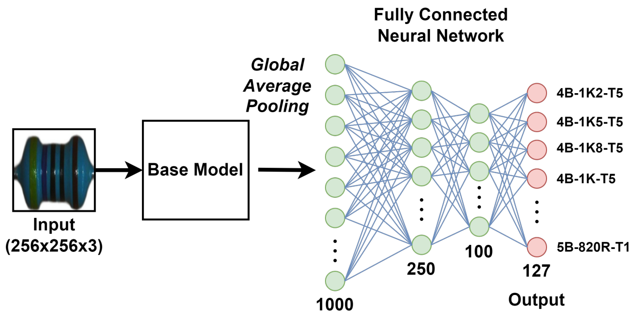

Figure 2: General Model Architecture (Dropout Layers not visualized)

(see Figure 2). Subsequent to this input layer, the base model is incorporated, which consists of a pretrained models offered by the Keras framework. These models may require a special rescaling of the input. The base model is then modified by excluding the top layers and replacing them with three hidden dense layers, with a ReLU activation function, and finally a softmax classifier with 127 neurons, thus completing the feed-forward neural network (FFNN). Following every dense layer (displayed as green circles), a dropout layer with a dropout rate of 0.6 is incorporated to mitigate overfitting. This value was heuristically determined.

To make up for missing training data and further minimize overfitting data-augmentation was applied. This involved randomly flipping the model horizontally and vertically, as well as modifying the brightness of the images.

The training process was then divided into two distinct stages. During the initial stage, the FFNN was subjected to training, while the base model was frozen. This preliminary training was terminated either when no further improvement in the validation loss was observed for 5 epochs, or when a maximum of 50 epochs had been reached. In the subsequent stage, some of the top layers of the base model were unfrozen, and the model was finetuned. The model was then trained for a maximum of 1200 epochs, with the patience of the early-stopping set to 50 epochs. The weights leading to the best validation accuracy were saved during the training process. For both stages, the Adam optimizer was used.

The training of all models was performed utilising a Nvidia GTX 1660. Due to TensorFlow's lack of native GPU support on Windows, the Linux Subsystem for Windows (WSL2) was used to enable GPU utilisation [9].

TABLE I: Specification of the (Base-) Model Architecture: Parameters, trainable Parameters and Size refer to the whole Model, while Accuracy [8], Depth and trainable Layers refer to the Base Model

Base-Model	Top-1 Accuracy ImageNet	Depth	Trainable Layers	Parameters	Trainable Parameters	Size
EfficientNetV2-S	83.9 %	513	500-513	21.9 mil.	2.5 mil.	83.54 MB
ConvNeXt-Tiny	81.3 %	133	120-133	28.9 mil.	10.5 mil.	110.16 MB

B. Model Specifics

Following the conception of a general training procedure, a series of base models were tested for the purpose of transfer learning. It was evident that some of these models significantly outperformed the rest. Therefore, the focus of this discussion will be on the results achieved by these models: the EfficientNetV2-S and the ConvNeXt-Tiny.

The EfficientNetV2-S is the smallest model of a CNN family formulated by Tan & Le in 2021. For their first generation of EfficientNets, Tan & Le created a new compound scaling method to effectively scale up CNNs. With this method, they achieved top classification accuracies while using far smaller and more resource-friendly models in comparison to state-of-the-art models. In their subsequent work, Tan & Le sought to enhance the training speed of these models while preserving their parameter efficiency. This was achieved by modifying the original architecture and proposing an enhanced progressive learning method, a strategy that led to the EfficientNetV2 surpassing the performance of earlier models while maintaining superior parameter efficiency and accelerated training times. [10], [11]

The second model to be considered is the ConvNeXt-Tiny, which is also the most compact model in its family. The ConvNeXt family was developed by Liu et al. in 2022, following the decline in the importance of CNNs due to the high performance of Vision Transformers (ViTs) in numerous vision tasks, which were introduced in 2020. By modifying the ResNet architecture with design elements found in ViTs, they were able to create a model family which surpasses comparable ViTs in many common computer vision tasks. [12]

The selection of both models was primarily driven by their high classification accuracy on the ImageNet dataset [8], while also exhibiting a reasonable number of parameters in relation to comparable models. Table I provides a detailed specification of the base models and the resulting transfer learning models. In the case of the EfficientNetV2-S and the ConvNeXt-Tiny, the training was exclusively focused on the top 14 layers of the base model. Despite the EfficientNetV2-S model being considerably deeper than the ConvNeXt-Tiny and having a significantly higher number of parameters in the FFNN, the ConvNeXt-Tiny model resulted in a model that was more memory-intensive.

C. Results

Following the discussion on the training process, the focus will now shift to the performance of the models. To this end, the accuracy, top-5 accuracy and interference time of each model has been measured. The interference time is only measured for one prediction, which renders the time unreliable. However, it serves the purpose of qualitative comparison. The measurement of results was conducted on a Thinkbook 15 G4 IAP, equipped with an Intel(R) Core(TM) i5-1235U 1.30 GHz Processor and 16 GB of RAM, without the use of a GPU.

A first evaluation of the models' performance yielded a test accuracy of 70.87 % for the ConvNeXt-Tiny and 77.17 % for the EfficientNetV2-S based model. Having trained two different models, the decision was taken to boost their accuracy by combining both models in an ensemble, i.e. by feeding the input to both models and averaging the output of both models [13]. This approach yielded a 7.08 % gain in accuracy compared to the previous EfficientNetV2-S. Furthermore, we experimented with flipping the model inputs and averaging their outputs, a technique comparable to the 10-crop approach proposed by Krizhevsky et al. [14]. This methodology demonstrated the highest accuracy for the EfficientNetV2-S model. By averaging the outputs from an unaltered image, the vertical flip, the horizontal flip and a combined flip, an accuracy of 86.61 % was achieved, representing an improvement of 9.44 % over the original performance. However, this approach is associated with significantly higher interference times. The application of this method to the ensemble did not enhance the top-1 accuracy. However, it did improve the top-5 accuracy by 3.15 %. The results of this study are presented in Table II.

V. Discussion

This study demonstrates the potential of machine learning models in classifying THT resistors. By designing a preprocessing pipeline, transfer learning was leveraged to successfully train models capable of identifying resistors based solely on their physical appearance. Two models achieved classification accuracies exceeding 70 %. Further improvements were realized through ensemble learning and input flipping techniques, culminating in a peak accuracy of 86.61 %. Notably,

TABLE II: Classification Results (v: +vertical flip, h: +horizontal flip & c: +combined flip)

Model	Accuracy	Top-5 Acc.	Runtime
ConvNeXt-Tiny	70.87 %	91.34 %	2.93 s
ConvNeXt-Tiny-v	74.02 %	92.91 %	3.70 s
ConvNeXt-Tiny-h	74.80 %	92.91 %	3.53 s
ConvNeXt-Tiny-v-h-c	78.74 %	92.13 %	5.90 s
EfficentNetV2-S	77.17 %	93.70 %	4.85 s
EfficientNetV2-S-v	**81.10 %**	**96.85 %**	11.89 s
EfficientNetV2-S-h	78.74 %	95.28 %	9.24 s
EfficientNetV2-S-v-h-c	**86.61 %**	**97.64 %**	15.38 s
Ensemble	**84.25 %**	95.28 %	5.73 s
Ensemble-v-h-c	**84.25 %**	**98.43 %**	20.18 s

the highest accuracy was achieved using flipped input images rather than an ensemble approach.

Despite these promising outcomes, several limitations were identified. The combined inference time of the best-performing model, approximately 20 seconds, presents a significant challenge for deployment on mobile devices. This issue could potentially be mitigated by incorporating traditional computer vision methods in the preprocessing pipeline or optimizing the classification model through techniques such as pruning to reduce its computational load [15].

Additionally, the reliance on substantial training data is a drawback compared to traditional methods that classify resistors based on their colour bands. An optimal solution would involve segmenting the colour bands and utilizing machine learning to identify the colours. As prior research by Muminovic et al. indicates challenges in identifying these bands, modern object detection models like YOLO could provide an effective solution for precise colour band segmentation.

For those interested in acquiring further knowledge regarding this project, please refer to GitHub: https://github.com/killphil/resistor_machinelearning [16].

References

[1] B. P. Engineering, “resistorcharts,” 2006. [Online]. Available: https://neurophysics.ucsd.edu/courses/physics_120/resistorcharts.pdf

[2] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[3] Pascal Niklaus and Gian Ulli, “Automated resistor classification,” 2015. [Online]. Available: https://pub.tik.ee.ethz.ch/students/2014-HS/GA-2014-02.pdf

[4] E. S. Mia Muminovic, *ICAT 2019: XXVII International Conference on Information, Communication and Automation Technologies : October 20-23, 2019, Sarajevo, Bosnia & Herzigovina : proceedings*. Piscataway, NJ: IEEE, 2019. [Online]. Available: https://ieeexplore.ieee.org/servlet/opac?punumber=8932650

[5] barrettotte, “Resistors,” 8/11/2023. [Online]. Available: https://www.kaggle.com/datasets/barrettotte/resistors

[6] Daniel Gatis, “rembg,” 12/30/2024. [Online]. Available: https://github.com/danielgatis/rembg

[7] R. Khanam and M. Hussain, “Yolov11: An overview of the key architectural enhancements.” [Online]. Available: https://arxiv.org/pdf/2410.17725

[8] K. Team, “Keras documentation: Keras applications,” 11/26/2024. [Online]. Available: https://keras.io/api/applications/

[9] “Install tensorflow with pip.” [Online]. Available: https://www.tensorflow.org/install/pip#windows-native_1

[10] M. Tan and Q. Le V, “Efficientnet: Rethinking model scaling for convolutional neural networks,” *International Conference on Machine Learning*, 2020. [Online]. Available: http://arxiv.org/pdf/1905.11946v5

[11] M. Tan and Q. Le V, “Efficientnetv2: Smaller models and faster training,” *International Conference on Machine Learning*, 2021. [Online]. Available: http://arxiv.org/pdf/2104.00298v3

[12] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A convnet for the 2020s.” [Online]. Available: http://arxiv.org/pdf/2201.03545v2

[13] O. Sagi and L. Rokach, “Ensemble learning: A survey,” *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, 2018.

[14] A. Krizhevsky, I. Sutskever, and Geoffrey E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, pp. 84–90, 2012.

[15] S. Yeom, P. Seegerer, S. Lapuschkin, S. Wiedemann, K. Müller, and W. Samek, “Pruning by explaining: A novel criterion for deep neural network pruning,” *CoRR*, vol. abs/1912.08881, 2019. [Online]. Available: http://arxiv.org/abs/1912.08881

[16] Philipp Bauerfeind, “Github: Resistor machine-learning,” 2024. [Online]. Available: https://github.com/killphil/resistor_machinelearning