# Final Project For ECE228 Track Number 1 Device State Classification with Images and Dynamometer Data

**Group Number 24: Eric Bressinger** **Nikhil Gandudi Suresh**

**Sharvari Satish Deshmukh**

## Abstract

In this project, we develop a lightweight, efficient, and scalable model to process multimodal data for real-time monitoring on edge devices. Our approach leverages a dual-pathway convolutional neural network architecture, integrating visual and time-series data to create a comprehensive feature representation. The visual domain pathway processes and extracts features from image data using depth-wise and point-wise convolutions, while the time-series pathway converts sequential data into 2D embeddings via Mel spectrograms, followed by convolutional processing. The features from both pathways are merged and passed through a temporal convolutional network (TempMixer block) to capture long-term dependencies. By employing depth-wise convolutions, we significantly reduce the computational load and model size, making it feasible for edge deployment. We validate our model on the Mudestra dataset, comprising image and dynamometer data from industrial milling machines, demonstrating substantial performance improvements through multimodal integration and data augmentation. Our experiments highlight the model's efficiency, with a size reduction from 200MB to 15MB, and its capability for parallel processing, making it ideal for real-time applications in resource-constrained environments. The code is at https://github.com/186shades/ECE228-Final-Project

I certify that I have filled out the evaluation.

## 1 Introduction

Industrial milling is a large industry heavily influenced by the quality of the milling bit. Determining the state of the milling tool is crucial for reducing inefficiencies in the milling process. The goal of creating a model to identify tool wear is to ultimately determine a Remaining Useful Life (RUL) metric for the tools. Most milling settings would not allow a large and computationally expensive model so there is a need for a model that is fast, small, and accurate. Ultimately, the model should be able to run on low-power or standby devices.

The "Multimodal Isotropic Neural Architecture with Patch Embedding" paper (8) uses a multimodal dataset called Mudestreda to classify a bit as sharp, used, or dull. There are two versions of the dataset, an original and an augmented dataset, which contains significantly more data. The model should be trainable with minimal data. The data types in the dataset include three-dimensional force signals recorded at 10 kHz and RGB images of the bit captured over five weeks. The paper employs two independent models for each data type, both of which are inaccurate on their own. The results from each modality are then concatenated and fed into a mixer model, resulting in a single model.

Preprint. Under review.

Our project's goal was to create a new model using similar techniques that is smaller in size and more accurate than the model in the paper. The primary focus of our project was to reduce size as much as possible to best fit a stand-by device application. We were able to achive a model accuracy of 98.2% with model size as small as 15MB.

## 2 Related work

Effectively integrating information from multiple modalities is a fundamental challenge in multimodal learning. Prior works have explored various fusion paradigms to combine multimodal data(1). These approaches often require manual design and are tailored to specific modality combinations like image and text. For applications involving time series or audio data, traditional single modality or self-supervised methods may not be sufficient due to data complexity and noise across modalities. Existing multimodal algorithms tend to be computationally expensive, limiting their applicability on resource-constrained devices. Several transformer based techniques have been proposed to optimize multimodal models, such as (5) (3) (4). All these methods come with high computational costs or reliance on large datasets. This also leads to models having millions of parameters and overfitting on small or medium multimodal datasets.

In contrast to prior work, the Minape framework proposed in the paper claims to be lightweight, efficient, and scalable for processing multimodal data on edge devices with limited computational resources. CNNs have been historically efficient at image classifications (2) especially in combination with data augmentation (6). By leveraging depthwise separable convolutions and isotropic convolutional neural architectures, Minape claims to reduce model size and latency while achieving high accuracy on small and medium datasets like Mudestreda.

## 3 Methodology

The goal is to train a model that is lightweight enough to be deployed on edge devices and fast enough for the devices/machines to be monitored in real time. These constraints determine design options for the model. As introduced before, multi-modal data is crucial to make a decision more confidently. This adds extra computing pressure on the edge device.

In the following proposed model, we aim to address all these issues while keeping the model simple and robust.

### 3.1 Unimodal Feature Representation

**Input Stacking**: The unimodal pathway takes $g_v$ tensors $T_j$ from a single modality, stacking them along their channel dimension to form tensor $T_v$ with dimensions $D^{h_v \times b_v \times d_v}$.

**Patching and Convolution**: Tensors are linearly patched to reduce internal resolution and then processed through isotropic channel-point convolutional blocks. This involves:

A single convolution layer with $d_{gv}$ input channels, kernel size $k$, stride $p$, and $c_{gv}$ output channels:

$$Q_v = BN\left(\sigma\{Conv(X_{i,j}, stride = p, kernelsize = k)\}\right) : D^{h_{gv}/p \times b_{gv}/p \times c_v}$$

Here, $\sigma$ denotes the Gaussian Error Linear Unit (GELU) activation function, which applies nonlinearity by weighting the input based on their percentile.

**Batch Normalization and Convolutional Blocks**: Post-activation, batch normalization (BN) normalizes each channel across mini-batch samples to reduce data variation.

Grouped Convolutions: Grouped convolutions, where groups equal the number of channels $c_v$, are used to facilitate channel-wise separable convolutions:

$$H_v = BN\left(\sigma\{DepthConv(H_v, groups = c_v\}\right) + H_v$$

Pointwise Convolutions: This follows the depth wise convolution, using a kernel size of 1x1:

$$H_{v+1} = BN\left(\sigma\{PointConv(H_v), kernel = 1 \times 1\}\right)$$

**Global Average Pooling**: The final layer is a global average pooling layer, which downsamples by averaging the vertical and horizontal dimensions of the input volume.

### 3.2 Visual Domain Pathway

The visual domain pathway processes visual data tensors through a series of normalization, embedding, and convolutional steps to extract meaningful features. Initially, tensors are grouped and normalized to fit the subnetwork dimensions. They are then patched and linearly embedded using a single convolution layer, followed by normalization with fast GELU and batch normalization. The resulting tensor is further processed through isotropic depth-wise and point-wise convolutions to produce the final feature vector $u_v$. This pathway's design ensures scalability and adaptability to different task complexities, allowing for precise and effective visual data representation.

**Tensor Grouping and Normalization**: Visual domain tensors are grouped into $G_v$ and normalized to ensure they meet sub-network dimension requirements. These tensors are stacked along their channel dimension to form a new tensor $T_v$ with a depth of $d_{gv}$:

$$\forall i = 1...L, \forall j \in G_v normv(T_{i,j}) : D^{h_j \times b_j \times d_j} \rightarrow D^{h_{gs} \times b_{gv} \times d_{gv}}$$

**Patching and Linear Embedding**: The normalized tensor $T_v$ is patched and linearly embedded using a single convolution layer. This involves:

$$Q_v = BN\left(\sigma\{Conv(X_{i,j}, stride = p, kernelsize = k)\}\right) : D^{h_{gv}/p \times b_{gv}/p \times c_v}$$

The convolution output is normalized using the fast GELU, followed by batch normalization (BN) as described in the equation:

$$H_v = BN\left(Q_v.\frac{1}{2}[1 + erf(\frac{Q_v}{\sqrt{(2)}})]\right)$$

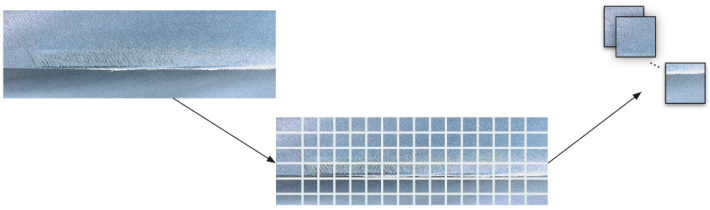

Figure 1: Patch Embeddings

**Isotropic Convolutional Blocks**: The tensor $H_v$ is fed into a sequence of isotropic depth-wise and point-wise convolutional blocks:

Grouped Convolutions: Grouped convolutions, where groups equal the number of channels $c_v$, are used to facilitate channel-wise separable convolutions:

$$H_v = BN\left(\sigma\{DepthConv(H_v, groups = c_v\}\right) + H_v$$

Pointwise Convolutions: This follows the depth wise convolution, using a kernel size of 1x1:

$$H_{v+1} = BN\left(\sigma\{PointConv(H_v), kernel = 1 \times 1\}\right)$$

**Global Average Pooling**: The final layer is a global average pooling layer, which downsamples by averaging the vertical and horizontal dimensions of the input volume.

The final output of the visual domain pathway is a vector $u_v$. The pathway's adjustable depth ensures that the architecture can scale according to the complexity of the task at hand, making it suitable for varying levels of detail required for different visual data processing tasks.

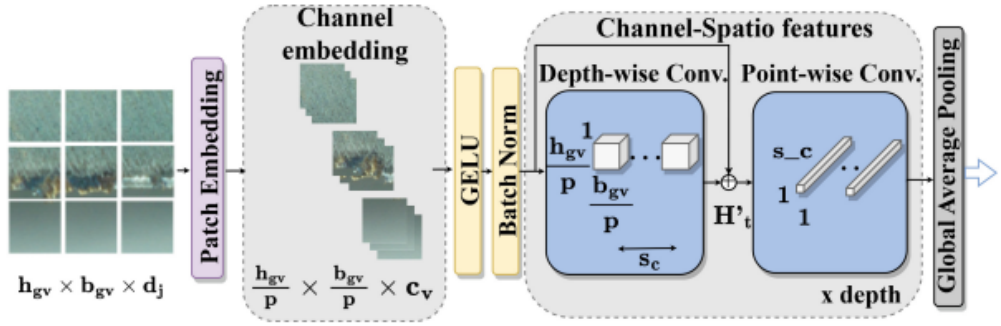


Figure 2: Visual Domain Pathway

### 3.3 Time Series Pathway

The process involves handling time series data through resampling, normalization, and transformation into two-dimensional embeddings using Mel spectrograms. These embeddings are further processed through convolutional neural network (CNN) blocks to capture and enhance feature representations.

**Grouping and Resampling**: Time series tensors are grouped into sets $G_s$ and resampled to a common length $n_{gs}$. The dimensions are adjusted (trimmed or padded) to match the required length $h_{gs}$.

$$\forall i = 1...L, \forall j \in G_s norms(T_{i,j}) : D^{h_j \times b_j \times d_j} \rightarrow D^{h_{gs} \times b_j}$$

**Two-Dimensional Embedding**: The core transformation involves converting the time series into 2D embeddings using spectrograms, which provide local frequency analysis of the signals. Short-time Fourier Transform is applied using a Hann window function to normalize signals, with a window size $n_{stft}$.

$$2D_{emb} = log10\left(STFT(input, stride, hann(n_{stft})\right)$$

Here, $hann(n_{stft})$ is the Hanning window function for STFT, and the stride size controls the sliding window step for STFT.

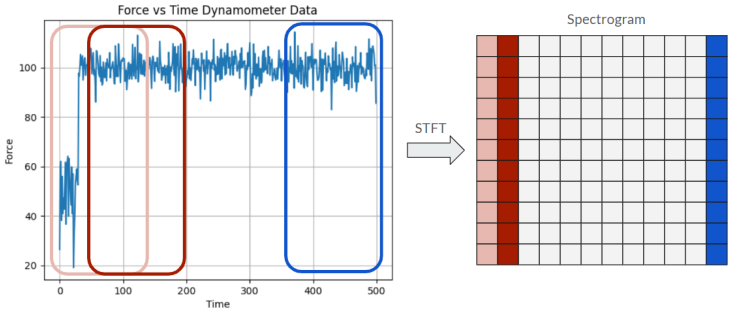


Figure 3: 2D Embedding of time-series data

**Stacking and Convolution**: The 2D embeddings of tensors in $G_s$ are stacked to form a new tensor $T_s$ with $d_s = \sum_{j \in G_s} b_j$ channels.

$$\forall j \in G_s, \forall k = 1...b_j T_s = 2D_{emb}[T_{:,j}(:, k)]$$

Next, the spectrograms are patched and embedded using linear operations, followed by processing through a channel-point convolution block, as described previously.

**Global Average Pooling**: The final layer is a global average pooling layer, which downsamples by averaging the vertical and horizontal dimensions of the input volume.

The final output of the visual domain pathway is a vector $u_t$.

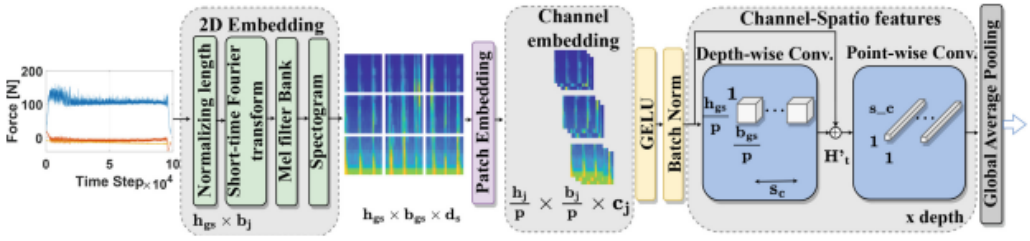


Figure 4: Time Series Pathway

### 3.4 Mixer Block

The learned representations from both the above pathways are concatenated to form a single joint representation vector u:

$$u = [u_v u_t]$$

This joint representation $u$ is passed to the mixer block, a temporal block designed to model long-term temporal dependencies in the data. The output of this block is $f_{temp}(u)$. $f_{temp}(u)$ is passed through a fully connected layer followed by a softmax layer to produce final predictions. While training the mixer block, the parameters in the time and visual pathways are frozen to enable faster learning in this block.

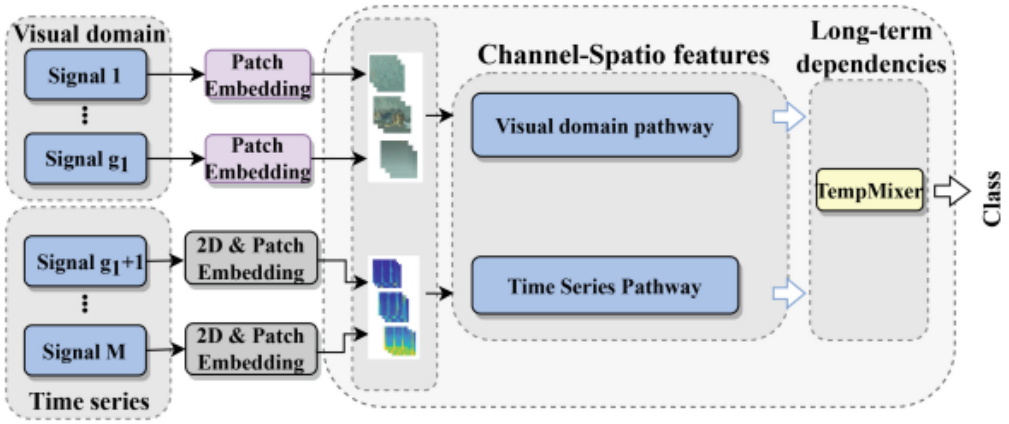


Figure 5: Complete Pathway

## 4 Experiments

We worked with Mudestra Dataset (7) which is obtained from a real industrial milling device with time series and image data. In the Mudestra dataset, there are four modalities to determine whether the blade is new, used, or dulled - one image, and three dynamometer measurements.

The dataset consists of 512 samples (4 modalities with three classes). Since, the dataset is quite small, we performed data augmentation (random rotation) to get a dataset of 3072 samples. We trained our model on both the original dataset and the augmented dataset to perform comparative analysis.

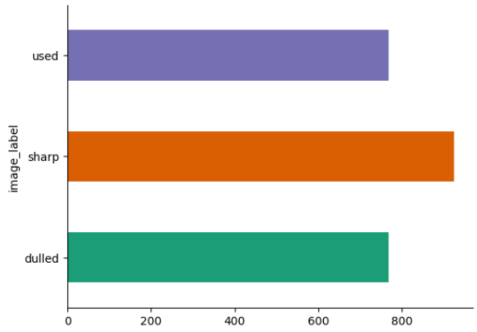


Figure 6: Class Distribution

All our experiments were performed for 10 epochs using Adam optimizer with a learning rate of 1e-3, and a batch size of 32.

### 4.1 Experiments on Original Data

The figures Figure 7, Figure 8, and Figure 9 display the loss and accuracy for the visual pathway, time-series pathway, and the mixer block respectively.

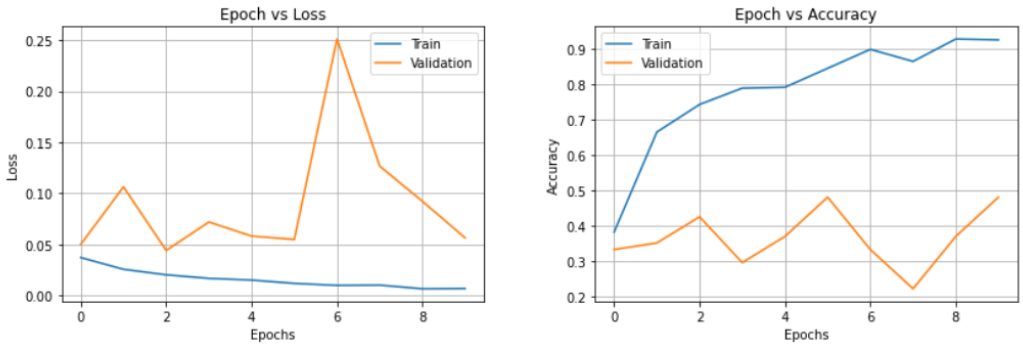


Figure 7: Visual Pathway Results on Original Data

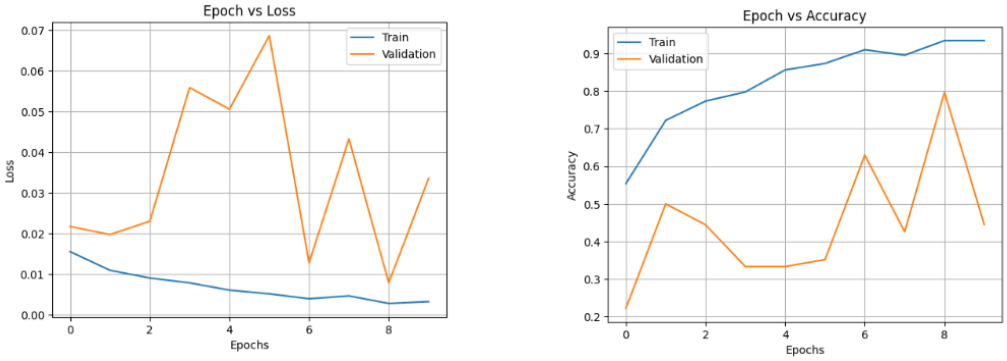


Figure 8: Time Domain Pathway Results on Original Data

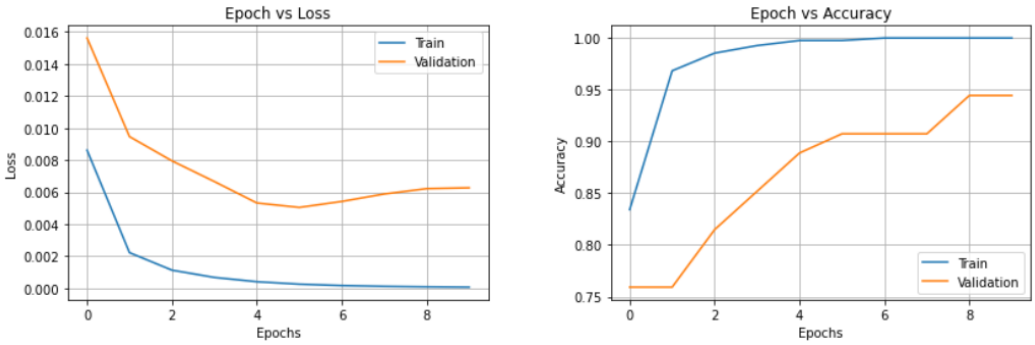


Figure 9: Results after the Mixer block using all the modalities on Original Data

### 4.2 Experiments on Augmented Data

The figures Figure 10, Figure 11, and Figure 12 display the loss and accuracy for the visual pathway, time-series pathway, and the mixer block respectively.

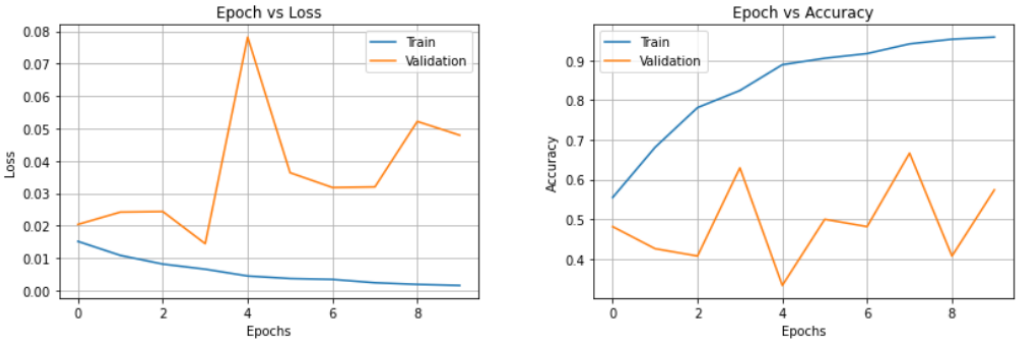


Figure 10: Visual Pathway Results on Augmented Data

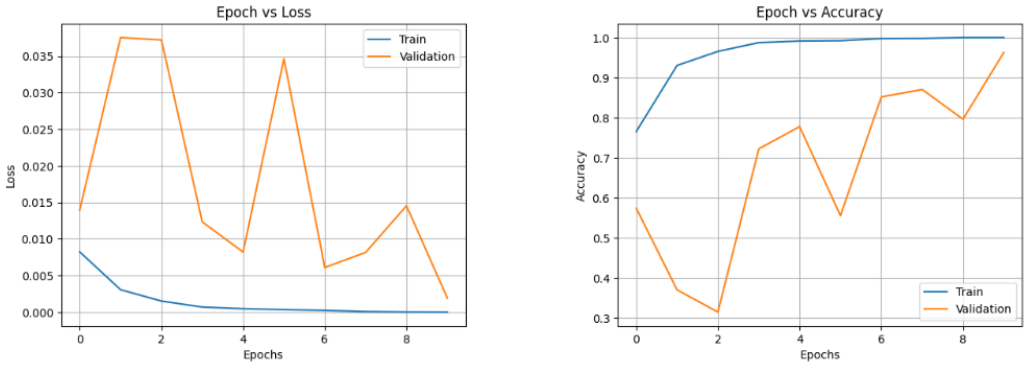


Figure 11: Time Domain Pathway Results on Augmented Data

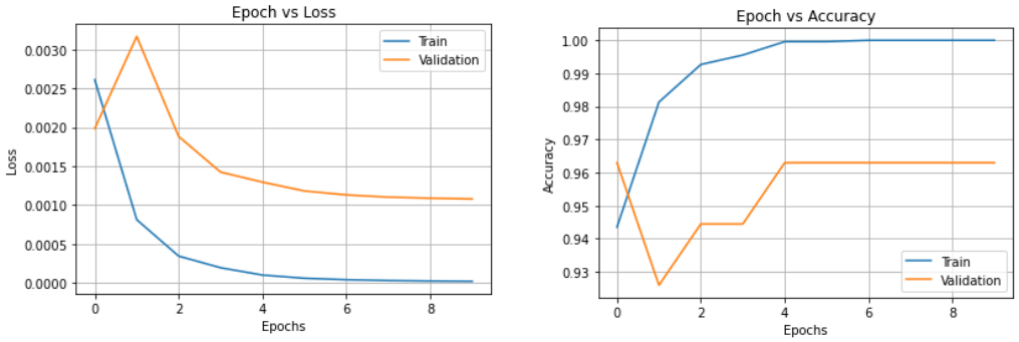


Figure 12: Results after the Mixer block using all the modalities on Augmented Data

### 4.3 Observations

- It can be noted that each of the modalities by themselves performs poorly. When the mixer block is used to combine results from all modalities, the performance improves drastically.
- Data augmentation acts as a regularizer and improves the performance of the individual models and the over all model respectively.

## 5 Conclusion

Combining multi-modal data efficiently for prediction or regression can be achieved using depth-wise convolutions, which significantly reduce computation and the overall size of the model. Our model using standard Conv2D operations is around 200MB, while a model employing Depthwise Conv2D followed by pointwise convolution is approximately 15MB. This efficiency allows the model size to grow linearly with the addition of new modalities, making it suitable for deployment on edge devices. Moreover, the approach is easily parallelizable, as each modality can run on its own processing chain, ensuring optimal performance and resource utilization.

## References

[1] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, 2019.

[2] L Chen, S Li, Q Bai, J Yang, S Jiang, and Y Miao. Review of image classification algorithms based on convolutional neural networks. *Remote Sensing*, 13(22):4712, 2021.

[3] Xinyu Gong, Sreyas Mohan, Naina Dhingra, Jean-Charles Bazin, Yilei Li, Zhangyang Wang, and Rakesh Ranjan. Mmg-ego4d: Multi-modal generalization in egocentric action recognition, 2023.

[4] Yan-Bo Lin, Yi-Lin Sung, Jie Lei, Mohit Bansal, and Gedas Bertasius. Vision transformers are parameter-efficient audio-visual learners, 2023.

[5] Xiaoyu Liu, Hanlin Lu, Jianbo Yuan, and Xinyu Li. Cat: Causal audio transformer for audio classification, 2023.

[6] C Shorten and TM Khoshgoftaar. A survey on image data augmentation for deep learning. *J Big Data*, 6(1):60, 2019.

[7] Hubert Truchan and Zahra Admadi. Mudestreda multimodal device state recognition dataset. `https://doi.org/10.5281/zenodo.8238653`, 2024. Zenodo.

[8] Hubert Truchan, Evgenii Naumov, Rezaul Abedin, Gregory Palmer, and Zahra Ahmadi. Multimodal isotropic neural architecture with patch embedding. In *International Conference on Neural Information Processing*, pages 173–187. Springer, 2023.