Project 2 - AI Thermal Fault Detection for Power Electronics

Designed By: Ishan Pandhare

I. Objective/Background:

A Python pipeline that:

- Generates or ingests thermal images of a power stage
- Labels them as Normal / Overheating / Fault
- Trains a model to classify them
- Outputs metrics
    - **Thermal image → operating condition**
- The model itself doesn't understand MOSFETs, inductors, or switching losses.
- It only sees temperature patterns.

What?

Electrical losses create spatial temperature patterns, faults distort those patterns, and neural networks learn to recognize those distortions.

Where operating condition =
✅ Normal
⚠️ Overheating
❌ Fault

In power electronics, temperature rises because of losses

1. Conduction Loss
    a. $P_{cond} = I^2 R_{DS(on)}$
        i. This heats MOSFET die and copper traces
2. Switching Loss
    a. $P_{sw} = \frac{1}{2} V I (t_r + t_f) f_{sw}$
        i. This provides localized heating near switching devices
3. Magnetic Losses
    a. Core loss
    b. Copper loss in inductors/transformers
        i. Applies more to hot inductors
4. Fault Loses
    a. Examples:
        i. **Shorted load** → massive MOSFET heating
        ii. **Gate failure** → shoot-through
        iii. **Component degradation** → asymmetric heating
5. Result
    a. These losses spatially distribute across the PCB:
    b. MOSFET zones get hot
    c. Inductor zones get hot
    d. Traces warm
    e. Sometimes one component spikes abnormally
    f. A thermal camera simply converts that spatial temperature field into pixel values.

Project 2 - AI Thermal Fault Detection for Power Electronics

Designed By: Ishan Pandhare

II. What this synthetic pipeline does

The baseline model demonstrates that physics-informed synthetic thermal data is sufficient to reliably detect and classify fault conditions in power electronics using convolutional neural networks, and physics.

Conceptually:

- I start with a board layout
- Example:
  - MOSFET at (x_1, y_1)
  - Inductor at (x_2, y_2)
  - Capacitor at (x_3, y_3)
- I define regions.
- Then I assign losses (Ex. from project 1, the buck converter project)
  - MOSFET: 3.2 W
  - Inductor: 1.1 W
  - Capacitor: 0.2 W
- Then, **losses are converted → to temperature**
  - $\Delta T = P \cdot R_{th}$
- Through this project, gaussian diffusion will occur to simulate conduction, so the hotspots of the image will be blurred, and this generates something that looks like a thermal image
  - These images come from loss distributions, not randomness

So, why is this powerful?

- Because faults don't just raise temperature.
  - They change thermal topology:
  - One device heats faster
  - One region spikes
  - Heat becomes asymmetric
  - Gradients sharpen
  - CNNs are excellent at spotting these spatial changes.
  - That's why thermal ML works.
- I am doing,
  - Loss-based thermal synthesis
  - Physics-guided data generation
  - Fault classification from spatial temperature fields
- Physics-informed machine learning for fault detection in power electronics.

Project 2 - AI Thermal Fault Detection for Power Electronics

Designed By: Ishan Pandhare

III. High-Level Overview Of Project

In this project, I am building a Python-based pipeline that uses thermal imaging to detect abnormal operating conditions in a power electronics system. The goal is to automatically classify system behavior as normal, overheating, or faulted by analyzing spatial temperature patterns, similar to how an engineer would interpret a thermal camera image.

Rather than starting with real thermal camera data, I generate synthetic thermal images based on estimated loss distributions in key components such as MOSFETs and inductors. These losses are derived from basic power electronics principles and then mapped spatially across a simplified board layout to produce realistic temperature fields. Each generated thermal image is labeled according to operating conditions, allowing me to rapidly create a large, controlled dataset while keeping the data grounded in physical behavior rather than random patterns.

Using this labeled dataset, I train a convolutional neural network to learn the characteristic thermal signatures associated with healthy operation, overheating, and fault scenarios. The model does not explicitly understand circuit topology or component physics; instead, it learns visual temperature patterns such as hotspot location, intensity, symmetry, and gradients. Once trained, the model can evaluate new thermal images and output probabilities for each operating state, enabling automated fault detection.

At a higher level, this project demonstrates how physics-informed data generation and machine learning can be combined to create an intelligent monitoring system for power electronics. The workflow mirrors real-world predictive maintenance: thermal measurements are analyzed continuously, abnormal behavior is identified early, and potential failures can be flagged before catastrophic damage occurs. This project emphasizes engineering-driven dataset design, model development, and system-level integration rather than treating machine learning as a standalone black box.

Short Version: I built an AI-based thermal fault detection system for power electronics using physics-informed data generation and machine learning. Synthetic thermal images are created from estimated component losses and used to train a convolutional neural network to classify operating states as normal, overheating, or faulted. By grounding thermal patterns in real loss mechanisms, the model learns meaningful temperature distributions rather than arbitrary image features. The system detects abnormal thermal behavior by analyzing hotspot location and intensity, demonstrating an automated approach to predictive maintenance and condition monitoring.

Project 2 - AI Thermal Fault Detection for Power Electronics

Designed By: Ishan Pandhare

IV. Documentation:

January 31 Progress:

The folder structure was set up locally and pushed into my github, and opened on vscode. Through vscode, I built a reproducible pipeline that generates a labeled synthetic thermal image dataset for power electronics, with class distinctions grounded in realistic loss-driven temperature patterns. It creates the data generation code, enforces a clean train/validation/test split, records metadata for traceability, and produces a visual montage to quickly verify that the thermal fault classes are visually and statistically separable.

II. Dataset Design and Generation

To enable controlled training and evaluation of thermal fault classification models, a synthetic thermal image dataset was generated. Synthetic data was selected to provide full control over class definitions, operating conditions, and fault characteristics while avoiding the scarcity and labeling challenges associated with real infrared datasets.

Thermal images were generated as grayscale temperature maps with spatial resolution of 256 × 256 pixels. Each image consists of an ambient background field combined with one to three localized hotspots representing major power components such as MOSFETs, diodes, and inductors. Hotspots were modeled as two-dimensional Gaussian distributions whose amplitude and spatial spread approximate relative power dissipation and thermal resistance effects. Additional noise and blur were applied to mimic infrared sensor artifacts and measurement uncertainty.

Three operating classes were defined. Normal samples exhibit moderate hotspot amplitudes and smooth thermal gradients representative of nominal operation. Overheating samples increase hotspot intensity and/or spatial spread to simulate elevated losses or reduced cooling effectiveness. Fault samples introduce abnormal localized patterns such as highly concentrated hotspots, asymmetric distributions, or secondary hotspots intended to represent conditions such as partial shorts, degraded thermal interfaces, or component failure.

The dataset was organized into training, validation, and test splits to support supervised learning and unbiased performance evaluation. All generation parameters and random seeds were logged to enable reproducibility and future dataset extension.

Update:

The dataset generator produced 3000 labeled thermal images split into train/val/test directories. A montage of representative samples was generated to visually verify class separability prior to training. All sample parameters and seeds were logged to labels.csv for reproducibility.

Project 2 - AI Thermal Fault Detection for Power Electronics

Designed By: Ishan Pandhare

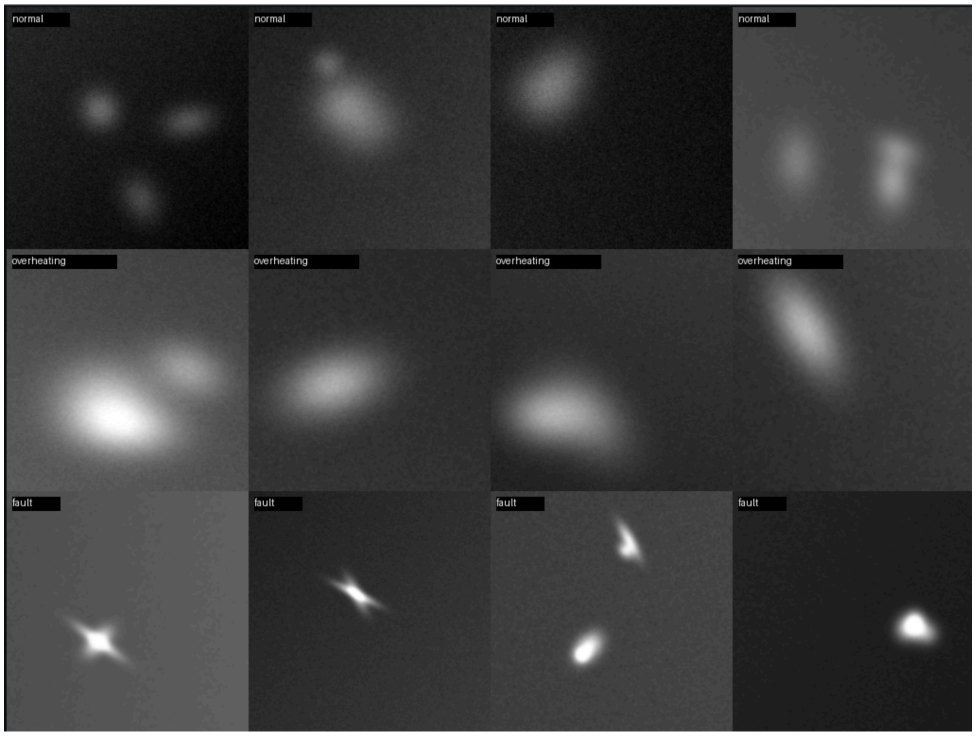


Figure 1. Thermal Fault Classification

Here is the sampled image that evaluate.py categorized by type of fault. This proves that the AI is able to carry out its task and sort out the faults based on its image characteristics.

February 6 Update:

In this phase, a physics synthetic thermal dataset was made which supports AI fault detection in power electronics. This is the ultimate project goal. Today bugs weren't just fixed, but a learning pipeline was unblocked and my system works now end-to-end.

The original issue was going to cause future errors/failures from

- Environment
- Dependancies
- Imports
- SSL/downloads
- Device issues

The ML pipeline is executable, not theoretical so the issues I resolved in the environment are

- Imports
- Dependencies
- SSL certificates
- Pretrained weight downloads

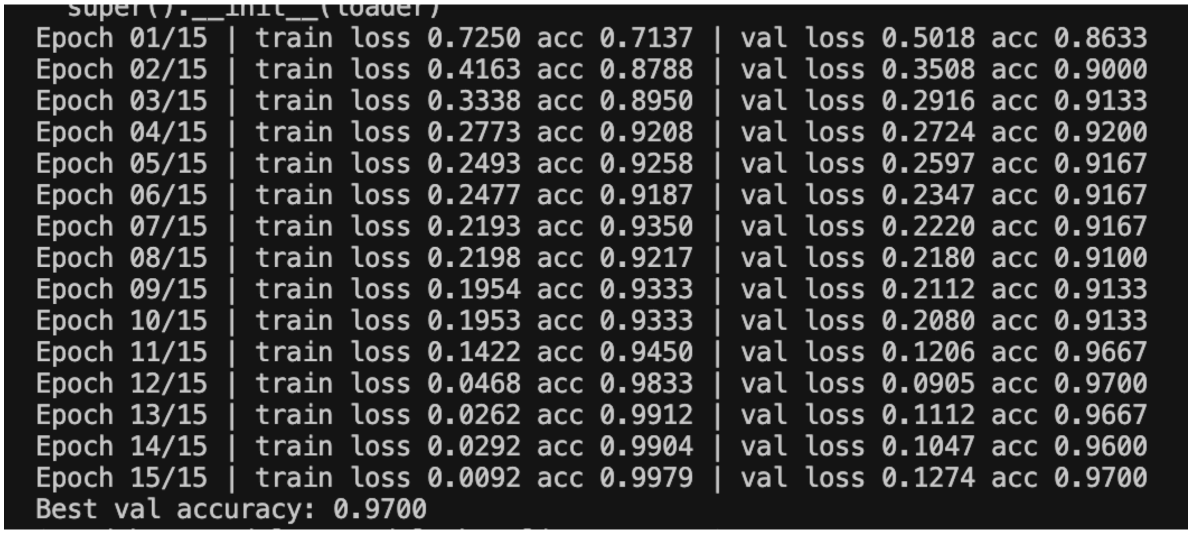
Now the code can run on a real machine with external dependencies like pytorch, torchvision, matplotlib, and are integrated properly. The training script can successfully initialize a model and start optimization.

- So the training started with epoch
- This shows the dataset generated is valid and the model architecture is compatible with the data
- The loss function and optimizer are behaving correctly
- A baseline is training now and everything else is becoming iteration and not risk

Project 2 - AI Thermal Fault Detection for Power Electronics

Designed By: Ishan Pandhare

This step was about getting to a stable baseline where the full data-to-model pipeline actually runs. Once training starts successfully, you know the dataset, model, and tooling are sound, and any further work is about improving performance rather than fixing setup issues.



```
    super().__init__(loader)
  Epoch 01/15 | train loss 0.7250 acc 0.7137 | val loss 0.5018 acc 0.8633
  Epoch 02/15 | train loss 0.4163 acc 0.8788 | val loss 0.3508 acc 0.9000
  Epoch 03/15 | train loss 0.3338 acc 0.8950 | val loss 0.2916 acc 0.9133
  Epoch 04/15 | train loss 0.2773 acc 0.9208 | val loss 0.2724 acc 0.9200
  Epoch 05/15 | train loss 0.2493 acc 0.9258 | val loss 0.2597 acc 0.9167
  Epoch 06/15 | train loss 0.2477 acc 0.9187 | val loss 0.2347 acc 0.9167
  Epoch 07/15 | train loss 0.2193 acc 0.9350 | val loss 0.2220 acc 0.9167
  Epoch 08/15 | train loss 0.2198 acc 0.9217 | val loss 0.2180 acc 0.9100
  Epoch 09/15 | train loss 0.1954 acc 0.9333 | val loss 0.2112 acc 0.9133
  Epoch 10/15 | train loss 0.1953 acc 0.9333 | val loss 0.2080 acc 0.9133
  Epoch 11/15 | train loss 0.1422 acc 0.9450 | val loss 0.1206 acc 0.9667
  Epoch 12/15 | train loss 0.0468 acc 0.9833 | val loss 0.0905 acc 0.9700
  Epoch 13/15 | train loss 0.0262 acc 0.9912 | val loss 0.1112 acc 0.9667
  Epoch 14/15 | train loss 0.0292 acc 0.9904 | val loss 0.1047 acc 0.9600
  Epoch 15/15 | train loss 0.0092 acc 0.9979 | val loss 0.1274 acc 0.9700
  Best val accuracy: 0.9700
```

Figure 2. ML Training Proof

Training is occurring

- Training loss is decreasing over time
- Validation accuracy is increasing as well
  - Highest validation accuracy is 97%
- Training curves are here

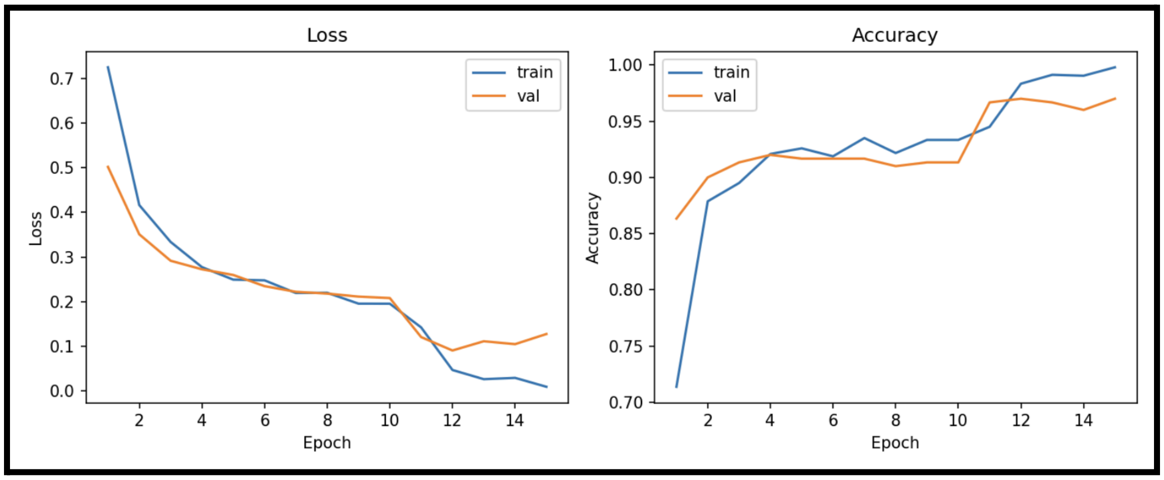


Figure 3. Plots of Loss and Accuracy of Training

Project 2 - AI Thermal Fault Detection for Power Electronics

Designed By: Ishan Pandhare

Training phase is complete, now moving on to evaluation

- **python3 -m src.evaluate --data_dir data --model_path models/baseline_resnet18.pt --batch_size 64**
- This gives us a test accuracy of **97.67%**
- Produces a confusion matrix as well

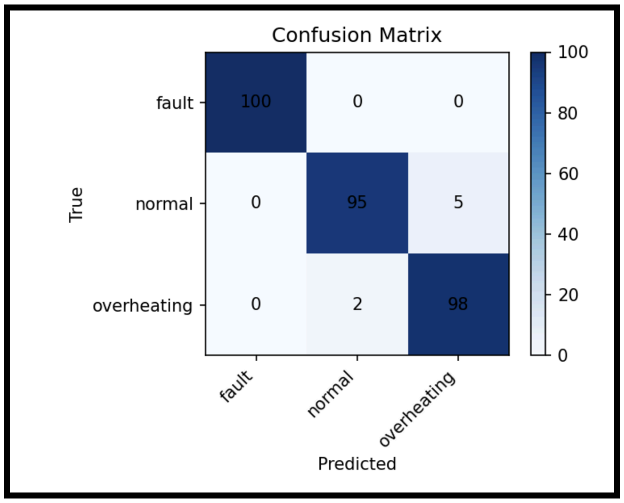


Figure 4. Confusion Matrix

Definition: "A confusion matrix is a tabular performance evaluation tool for machine learning classification algorithms that displays the count of correct and incorrect predictions, broken down by class"(). In this case the confusion matrix demonstrates the accuracy on performance evaluation according to the classifications specified for operating classes; fault, normal, overheating.

- **Test metrics output**
  - class,precision,recall,f1,support
  - Fault, 1.0, 1.0, 1.0, 100
  - Normal, 0.9793, 0.95, 0.9644, 1.00
  - Overheating, 0.9514, 0.98, 0.9655, 1.00
- **Per-class metrics** (why these are strong)
  - **Fault**
    - Precision = 1.00
    - Recall = 1.00
    - No false positives, no false negatives
  - **Normal**
    - Slightly lower recall
    - Some borderline overheating cases look "normal" thermally
  - **Overheating**
    - High recall and precision
    - Model captures distributed heat patterns well

Project 2 - AI Thermal Fault Detection for Power Electronics

Designed By: Ishan Pandhare

V. Description of Results:

The trained baseline convolutional neural network achieved a test classification accuracy of 97.67% on the held-out dataset. The resulting confusion matrix indicates perfect detection of fault conditions, with all fault samples correctly classified. Limited misclassification occurred between the normal and overheating classes, which is expected due to their overlapping thermal characteristics and gradual transition between operating regimes.

Per-class evaluation metrics further support this behavior. The fault class achieved a precision and recall of 1.00, indicating no false positives or false negatives. The normal and overheating classes maintained high precision and recall values (>0.95), confirming that the model reliably distinguishes distributed thermal patterns associated with elevated losses. Overall, these results demonstrate that the synthetic dataset encodes physically meaningful thermal features and that the trained model successfully learns class-discriminative representations relevant to power electronic fault detection.

VI. Limitations and Next Steps

Though this is a very compelling project that can apply to real life devices, some limitations include the synthetic data that does not fully capture the scope of real-world emissivity effects along with a simplified board layout. Ambient temperatures are assumed to be uniform also which differ in real settings. Some great next steps and improvements I can implement would be

- Real IR data
- Domain adaptation
- Explainability