0.1 Introduction

The interset for aureora follows to some exdent the solarcykel which menas that the during large activits in the sun the probabilty for auriora vents increarses. And the incres possibilty for autora events pushes the inters amopng not scientific citisenst to step outsid in the cold and dark to se the aurora if we consider the borelis. The groving interst for aurora during solar-cykle paek activuty, does not only gives the oportynity for a the community to explore an amaizing color explosion in the night sky, but can also be the connection to a more scienfic aproch to the phenomene and an eager to learn more about the space weather and the sun-earth connection.

In order to maximize the chances of observing aurora, a special designed aurora detector have been created [1]. This detector uses firstly light sensors to measure the light intensity in the sky, and secondly temperature and humidty the estimate cloud cover. An algorithm have been developed to combine the data from these sensors to provide an aurora probability index.

0.1.1 Deployment requirements on aurora detector

In the light of aurora detector a natural selection of deployment of the physical device would be in remote areas with low light pollution and minimal human activity. However, since the teh overall idea of the device is to alert neigbourging citizens that an aurora event is happening, the device should not be deployed in areas where no one lives. Therefore, the device should be able to handle various weather conditions, including frost, snow, rain, and arbtray artifical light at least to some extent.

This restricts the posibilties of deployment locations to in side the urban areas or suburban areas where the device can be placed on rooftops or in antenna toweres.

The main motivation for the project [1] was to create an affordable aurora detector that could be deployed in larger numbers close to populated areas to increase the chances of citizens observing aurora events. This means that both the cost of the device and the cost of deployment should be kept low.

Considering the deployment cost, the device should be selfsustained in terms of power consumption, meaning that it should use a solar panel and a battery to store energy, this implies no extra cost for power supply.

It is also consieder that device will have WiFi connection sicne that is a common feature in urban and suburban areas, this will enable the device to send alerts the community.

0.1.2 Motivations for applying Deep-Learning model on Aurora Detector

The main motivation for applying deep-learning models to the aurora detector is to enahnce the detection of aurora events while keeping the false positive rate low. The current algorithm, while effective, have stille some issues when frost or snow is present on the sensor, leading to false positives.

In [5], [7] the point out the the advance of using edge Ai, e.i . deploying machine learning models on microcontrollers, to enable real-time data processing and decision-making directly on the device and the shortcomming.

In [6] the menthon the limitation in bandwidth which might not be a big issue for the aurora detector, deployed in urban or suburban areas with WiFi connection, but still a point to consider. It have been shown that most of embedded device power consumtuin comes from the transfering of data [4].

In [7] they emphesice the aspect of latency, which is a important factor for the aurora detector since the goal is to alert citizens in real-time when an aurora event is happening. Getting an allert a few minutes late might result in missing the event.

Both [5] and [7] highlights the privacy aspect of edge AI, since data is processed locally on the device, there is less risk of sensitive information being exposed during data transmission. However, this is not a major consern regarding data for the aurora detector.

Lastly the main motivtation for using deep-learning models on the aurora detector is the potential for the model to reject false positives caused by frost, snow, and other environmental factors, but still have an accuracy as good as or better than the current algorithm [1].

0.2 Electronics overview

This section is fully described in [1] and here follows a short overweivew of the electronics used in the aurora detector.

0.2.1 Microcontroller

The current aurora detector uses an ESP-8266 deployed on a developer board from Wemos, Wemos D1 MINI. The microcontroller is used to read data from the sensors, process the data using the aurora detection algorithm, send the raw data and aurora probability index to a server via WiFi. The only up and running device is powered by a 5V power supply continously.

Since the goal is to deploy deep-learning models on the aurora detector, the microcontroller must be replaced by a more powerful one, capable of running the deep-learning models. The device considerd in this project is a Raspberry Pi Pico W, which is a microcontroller with WiFi capabilities and enough processing power to run deep-learning models using TensorFlow Lite for Microcontrollers.

0.2.2 Light sensor

There are two TSL2591 lightsensors deployed on the aurora detetcter. In front of one of the sensors a lulty layer interference filter is applyd filtering freequncies close to 557 nm (green), whcich is the most prominent frequncy emitted during aurora events. The sensorsn returns values between 0-65535, in given gain, representing the light intensity measured by the sensor, from two different channels with slightly different spectral responses, one more agaibt the IR spectrum and one closer to the visual spectrum.

0.2.3 Infrared thermometer

The infrared thermometer is a MLX90614 which is used to measure the "temperature" of the sky. The basic idea behind the sensor is that a clear sky will have a lower temperature compared to a cloudy sky, since clouds emit infrared radiation. The ssensor returns a temperature value in celsius between in floating point representation.

0.2.4 Humidity and temperature sensor

The humidity and temperature sensor is a DHT22 which is used to measure the ambient temperature and humidity. The main purpose of this sensor is that the values obtained can be used to calculate for a given humidty and temperature what the clear sky temperature should be. This can then be compared to the measured sky temperature from the infrared thermometer to estimate cloud cover.

0.3 Data collection and preprocessing

The data collected for this project was conducted between 2024-12-01 and 2025-11-30 from a device deployed in Mora , Sweden (61.0°N, 14.5°E). The data is collected from a poblic avalaiable Thingspeak instance [2], eventhough administartor access is required to download the full dataset.

The raw dataset contains the following feautures:

$\text{created}_at : Timestamp of when the data was recorded. entry_id : Unique identifier for each data entry. Field1 :$
$Light intensity from the filtered TSL2591 sensor (green light). Field2 : Light intensity from the unfiltered TSL2591 senso$

$Calculatedonmicrocontroller, clearskyvaluebasedondatafromMLX90614andDHT22Field4 : Calculatedonmicrocontr$
$20basedonanalgorithmcombiningdatafromallsensors.Field5 : TemperaturefromDHT22sensor.Field6 :$
$HumidityfromDHT22sensor.Field7 : SkytemperaturefromMLX90614sensor.Field8 : TheIR-$
$channelfromtheTSL2591sensorwithoutafilter.$

Since the data had an median sampling rate of 1/15 Hz, the the full year data contaned approximately 2 million samples. To save some computational power a algrothm was applied to calculate if its day or night that takes current date time and location into account. This algorith is currently operationg as an veto on the current aurora device, but this information is not sent to Thingspeak. This alogrithm was applied to the full dataset to filter out day-time samples, since visuable aurora events only occur during the dark hoyurs this reduced the data set by approximatöy half.

To capture frost add on and frost removal two additional features were added to the data set. Since humidity along side with the temperature device capture to large extend the posiblity for frost to occur, a new feature called "Rolling Mean Temperature (C)" and "Rolling Mean Humidity (%)" where added to the dataset. The rolling mean was calculated using a window size of 20 minutes with the previous 20 minutes data. Endpoints were handled such that only samples within a full window were considered. The idea was that these two values could help the model to learn when frost is likely to occur and when it is likely to melt away.

Since in this project goal is to apply deep-learning models on the aurora detector, it was considerd that most of the precalculations done on the microcontroller should be avoided, to let the model learn directly from the raw sensor data. Therefore, only the following features were used for training the models:

Filter 557nm, Light intensity from the filtered TSL2591 sensor (green light). No filter,Light intensity from the unfiltered TSL2591 sensor. Temperature (C), Temperature from DHT22 sensor. Humidity (%), Humidity from DHT22 sensor. Sky Temperature (C), Sky temperature from MLX90614 sensor. IR, The IR-channel from the TSL2591 sensor without a filter. Rolling Mean Temperature (C), Rolling Mean Temperature (C) over 20 minutes. Rolling Mean Humidity (%), Rolling Mean Humidity (%) over 20 minutes.

0.3.1 Labeling strategy

From observations between 2024-12-01 and 2025-11-30 there were in total 8 aurora events visible from the deployment location verified by observer over 8 different dates. One verified event with no aurora, eventhogh the algorithm on the microcontroller indicated a high aurora probability, was also observed.

To label the data, a time window sweeping over the data from beginning of considered aurora event to end of event was applied. The time conseiderd was between 2 and 6 hours. The windows was sweeped over the data with in the consederd time at one verified dates and based on visual inspection of the light sensor with filter at 557nm. Most of the data points including a value from 557 nm sensor having a larger value than 3 was considered as aurora event. The threshold value of 3 was chosen based on visual inspection of the data from all verified aurora events.

Addinal labeling was done for the known non-aurora event with the same priciples mentioned previoulsy.

Moreover, 5 dates with high aurora points from the microcontroller algorithm, but not confirmed by obserer, where also consederd. Similar procedure were applied as above, but in order to avoid mislabeling of aurora events, a check of the magentometer located in Kiruna and Lyksele was used to varify the aurora event. Only those events with high magnetometer activity, and aurora points larger than 3 was considerd as aurora event.

0.3.2 Data summary

After filtering out day-time samples and applying the labeling strategy, the final dataset used for training and evaluating the deep-learning models contained approximately 1.8 M samples with unlabeled data and approximately 14,000 labeled samples, where around 13,000 samples

were labeled as non-aurora and around 1,000 samples were labeled as aurora. This indicates a significant class imbalance, with non-aurora samples vastly outnumbering aurora samples.

Before training with data a normalization of the data was applyed using the mean and standard deviation caluclated from the training set. The normalization was done using the formula:

$$X_{norm} = \frac{X - \mu}{\sigma}$$

where X is the original feature value, and μ and σ are the mean and standard deviation of the feature in the training set, respectively.

0.4 Model development

Since the perpose of this project is to deploy deep-learning models on microcontrollers, the model architecture should be kept simple to reduce the computational cost. And for making prof of priciples a simple fully connected neural was created.

Since the dataset contained a lot of unlabeled data and just a small fraction of labeld data and for that matter highly un balenced, a semi-supervised learning approach was considerd, which in this context means that the models have been trained on unsuperviced data first and than part of the model fine-tuned on the labeled data.

This project follows to large extent the guidelines for that type of training from [3]. The models are created by Keras and the script describing the model creation and training is written in Python and can be found in the GitHub repository 'tiny$_m l_c ode/models/FC_a utoencoder.py'[**alinHansAlinTinymlauroradetect**

0.4.1 Model architecture Autoencoder

The model architecture used in this project is a simple fully connected neural network with the following layers:

- Input layer: 8 neurons (one for each feature)
- Encoder Hidden layer 1: Dense layer with 8 - 256 neurons, Leaky ReLU activation
- Encoder Hidden layer 2: Dense layer with 8 - 128 neurons, Leaky ReLU activation
- Encoder Latent layer: Dense layer with 2 - 64 neurons, Linear activation
- Decoder Hidden layer 3: Dense layer with 8 - 128 neurons, Leaky ReLU activation
- Decoder Hidden layer 4: Dense layer with 8 - 256 neurons, Leaky ReLU activation
- Decoder Final layer: Dense layer with 8 neurons, Linear activation

Regarding the model functioning as a classifier the model inherits the encoder part from the autoencoder model and adds the following layers:

- Classifier Hidden layer 3: Dense layer with 8 - 128 neurons, Leaky ReLU activation
- Classifier Last layer: Dense layer with 1 neuron, Sigmoid activation

0.5 Training procedure

Two types of of classifying models were coinstructe and trained in this procjetc, one using only the labeled data and one using semi-supervised learning approach.

0.5.1 Semi-supervised learning

The semi-supervised learning approach involved two main steps, pretraining the autoencoder model on unlabeld data and fine tuning the model on labeled data.

The autoencoder were able to train in 200 epochs but a callback with early stopping was used to monitor the validation loss with a patience of 10 epochs. When the model were trained the weights were saved.

The classifier model "Encoder-Classifier" inherith the same archteicture as the encoder part from the previously mentioned autoencoder but with additional classifier layers. The weights from the encoder part were loaded from the trained autoencoder model.

Initially the weights from the encoder part were frozen as mentioned in [3] and the mdoel had the opurtunity to train for 200 epochs. However, the erlystopping callback prevented the model to get overfitted. When the first time in thsi classifiear training were completed, either from early stopping or when the model reache 200 epochs, the weights from the encoder part were unfrozen and the whole model were trained again for 200 epochs with early stopping.

Bibliography

[1] Hans Alin. *Aurora Detector : Affordable Device for Detection of Optical Aurora*. 2022. URL: https://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-114622 (visited on 01/12/2026).

[2] *Aurora Detector - ThingSpeak IoT*. URL: https://thingspeak.mathworks.com/channels/1636584 (visited on 01/12/2026).

[3] Aurélien Géron. "Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow". In: ().

[4] Benjamin Karic et al. *Send Less, Save More: Energy-Efficiency Benchmark of Embedded CNN Inference vs. Data Transmission in IoT*. Oct. 30, 2025. DOI: 10.48550/arXiv.2510.24829. arXiv: 2510.24829 [cs]. URL: http://arxiv.org/abs/2510.24829 (visited on 01/12/2026). Pre-published.

[5] Swapnil Sayan Saha, Sandeep Singh Sandha, and Mani Srivastava. "Machine Learning for Microcontroller-Class Hardware: A Review". In: *IEEE Sensors Journal* 22.22 (Nov. 15, 2022), pp. 21362–21390. ISSN: 1530-437X, 1558-1748, 2379-9153. DOI: 10.1109/JSEN.2022.3210773. URL: https://ieeexplore.ieee.org/document/9912325/ (visited on 11/15/2025).

[6] Daniel Situnayake and Jenny Plunkett. *AI at the Edge*. " O'Reilly Media, Inc.", 2023. URL: https://books.google.com/books?hl=sv&lr=&id=l6imEAAAQBAJ&oi=fnd&pg=PR4&dq=+%22AI+at+the+Edge%22&ots=gaD3yAiBv2&sig=jZOpW7vjIHQxXu66qb8y0Bf8Mxc (visited on 01/12/2026).

[7] Pete Warden and Daniel Situnayake. *Tinyml: Machine Learning with Tensorflow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, 2019. URL: https://books.google.com/books?hl=sv&lr=&id=tn3EDwAAQBAJ&oi=fnd&pg=PP1&dq=Machine+Learning+with+TensorFlow+Lite+on+Arduino+and+Ultra-Low-Power&ots=jqrp9t88t1&sig=KY2N3QxM6r43qbqUaohhKRrKOao (visited on 01/12/2026).