Comparative Analysis of Deep Learning Architectures for Solar Flare Prediction from Multi-Temporal Image Sequences

Antonio Lissa Lattanzio[1]

Abstract

In this paper, we present a novel approach for predicting solar flares using a combination of deep learning models and line-of-sight (LOS) magnetogram images. Our goal is to predict solar flare events occurring within 24 hours from a sequence of three consecutive magnetograms. We propose three different deep learning architectures: Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks and Vision Transformers (ViTs), to extract and analyze features from the image sequences.

Class imbalance is mitigated through weighted sampling and sliding-window cross-validation on temporal data. The results show that all models have the capacity to predict flare activity effectively, while validation set balance remains a key factor for stable generalization.

Keywords

solar flare forecasting, deep learning, CNN, LSTM, ViT

1. Highlights

- Developed a new dataset of full-disk line-of-sight (LOS) magnetograms sampled at 72h, 48h, and 24h before flare events.
- Tested multiple CNN-LSTM architectures and ViT architecture to choose the best performing model.
- Introduced a novel method of concatenating multi-temporal magnetograms for input into a Vision Transformer (ViT), in order to model temporal relationships via spatial self-attention.
- Designed a custom time-series K-Fold validation where train, validation, and test sets preserve chronological order to simulate real-world prediction conditions.
- Implemented Grad-CAM explainability analysis to visualize spatial attention and verify that model focus aligns with physically relevant solar active regions.

2. Introduction

Solar flares are sudden bursts of electromagnetic radiation from the solar atmosphere, often in the extreme ultraviolet and X-ray regimes. They are classified on different categories based on the peak X-ray flux. These categories are, from less to most powerful: A, B, C, M, X (Sinha et al. [1]). The magnetic fields generated in the Sun generates sunspots, the concentrations of strong magnetic field, that appear as dark spots. Occasionally, the magnetic energy stored around the sunspots is released generating a solar flare. Even though solar flare events did not cause disasters in the past, modern society is becoming more and more susceptible to space weather as it relies more and more on space assets and large-scale power grids, which suffer from such disturbances generated by this phenomenon (Isobe et al. [2]). Thus, being able to predict these events is becoming more and more important.

In recent years, advances in deep learning have introduced new possibilities for flare forecasting. Deep learning models, especially Convolutional Neural Networks and Recurrent Neural Networks, have shown good capabilities of understanding spatial and temporal patterns in image sequences. Many studies have proposed CNN or CNN-LSTM-based architectures using data from instruments like the Helioseismic and Magnetic Imager (HMI) [3] or derived datasets such as SHARP (Space-weather HMI Active Region Patches) [4] . However, these approaches typically focus on either static snapshots of active regions or summary parameters, rather than the direct use of images that show the evolution of the full disk Sun surface.

This paper aims to fill the gap by proposing a comparative study of deep learning architectures for solar flare prediction using multi-temporal full-disk line-of-sight magnetograms. Specifically, we evaluate the performance of three distinct architectures: (1) a baseline CNN-LSTM model that combines spatial feature extraction with temporal sequence learning; (2) a pre-trained encoder-decoder CNN integrated with an LSTM layer; and (3) a Vision Transformer (ViT) model designed to learn both spatial and temporal relationships through self-attention mechanisms. Our approach uses sequences of LOS magnetograms sampled at 72, 48, and 24 hours before a flare event to predict the occurrence of a strong solar flare (M/X class) in the next 24 hours.

One of the biggest problems when dealing with solar flare forecasting is the heavy unbalance of the classes, in particular we have that low energy flares (A, B and C) are way more likely to happen compared to strong flares (M and X), this is a problem for deep learning approaches since it will make the models to be biased towards the majority class. To mitigate this problem we tried to make the dataset less unbalanced compared to the ones that have already been used in the related literature by undersampling the low class and using class-weights in the loss function when training the models.

We developed a new dataset of full-disk magnetograms taken from the Helioviewer API, ensuring time consistency, quality control, and trying to reduce the class unbalance for training and evaluation. Unlike many prior studies that rely heavily on summary statistics or handcrafted features, our models are trained directly on the magnetogram images, which contain richer spatial information. We also introduce a custom time-series K-Fold validation approach to ensure that the models are evaluated under realistic forecasting conditions, preserving chronological order and realistic forecasting conditions.

Our contributions are three: (1) we propose and compare three deep learning architectures for solar flare prediction

using temporally sequenced LOS magnetogram images; (2) we construct a novel dataset for this task and address the challenges of data imbalance without relying on traditional augmentation; and (3) we test different kind of models to find the best performing one.

3. Related Work

In the recent years there have been many attempts to predict solar flares events using deep learning approaches. There are two main approaches when dealing with predicting flares using images: the first method uses Helioseismic Magnetic Imager Active Regions Patches (SHARPs; Bobra et al. [4]) which are derived from the full-disk magnetograms of the Helioseismic Magnetic Imager (HMI; [3]), the database contains maps and summary parameters of automatically tracked active regions of the Sun from 2010 to the present day. The second is based on full-disk line-of-sight magnetograms from the Helioseismic and Magnetic Imager. The data from HMI are more recent and of higher quality but there's few sample of strong solar flares. This is why many studies used also magnetograms from the Michelson Doppler Imager (MDI; Scherrer et al. [5]), which was active from 1996 to 2011. Compared to it's successor HMI the MDI magnetograms are not as good in quality (Sun et al. [6]).

In the prediction of flares using active regions Sun et al. [6] developed an ensamble architecture composed of CNN together with an LSTM, where the CNN uses static magnetograms before the prediction period of 24h and the LSTM uses summary parameters of 24h long time series before the prediction period. Zang et al. [7] proposed an architecture based on Convolutional Neural Network model with one-against-rest approach that consist in training 4 binary models, one for no flare class (class N that consists in A and B flares) vs CMX, one for C vs NMX, one for M vs NCX and one for X vs NCM. The work of Grim et al. [8] propose a transformer-based architecture to forecast $\geq M$ class flares, using sequences of line-of-sight magnetogram images as input in order to observe the evolution of active regions, this work also used data augmentation to deal with the class imbalance. Sun et al. [9] developed two 3 dimensional CNN to predict whether an AR would erupt $\geq C$ and $\geq M$ within the next 24 hrs, the aim of 3D CNN is to study toghether the temporal and spatial evolution of the sunspots.

In the full-disk line-of-sight prediction Yan et al. [10] developed a deep learning forecast system comparing four different CNN architecture and ViT for predicting $\geq C$ and $\geq M$ flares in the next 24h. Park et al. [11] compared three CNN architectures in the prediction task of $\geq M$ flares. Kaneda et al. [12] proposed a Flare Transformer, which handles time series of images and physical features to produce predictions on 4 classes.

There are also approaches that do not use images but deal only with the temporal evolution of summary parameters calculated from the magnetograms like the work of Liu et al. [13] that developed three LSTM models to predict $\geq M5.0$, $\geq M$, and $\geq C$ class that use 40 features; a first group of 25 physical parameters that characterize AR magnetic field properties and a second group of 15 features related to flaring history. The work of Chen et al. [14] used K-Nearest Neighbors, Random Forest, and XGBoost to predict the total flare index and the maximum flare index of an active region within the subsequent of 24, 48, and 72 hrs using four parameters calculated by the magnetograms.

Based on the related work we found that there are no studies about predicting solar flare by studying the evolution of the magnetograms, this is why in this paper we propose three architectures for predicting solar flares based on a time series of 3 magnetograms at 24, 48 and 72 hours before the flare event using two CNN-LSTM architectures and a vision transformer, with this work we aim to build a model that can improve the performances by studying together the spatial and temporal evolution of full-disk line-of-sight magnetograms.

4. Data

4.1. Data Acquisition and Pre-processing

Our objective is to train models that predict the solar flare class based on full line-of-sight magnetograms of the Sun's surface. The data was obtained using the SunPy library [15], focusing on solar flares ranging from class A to class X, recorded between 1996 and 2024. The collected data was transformed into a pandas DataFrame for further analysis.

The retrieved annotations for the classes A, B, C, M and X was: 101, 18187, 28797, 3555, 251.

During inspection, it was observed that multiple annotations existed for each day, ranging from 1 to over 30. To address this problem, we retained only the annotation with the highest class for each day. The number of annotations dropped to: 86, 2197, 3719, 1594, 209. The class unbalance is clearly visible, since, as the other studies did, we want to build a dataset of strong (M, X) vs quiet (A, B, C) solar flares, we mitigated the unbalance by randomly sampling 1600 events for the classes B and C.

Magnetograms are usually provided by the JSOC service[1], but at the time their servers were down, thus we queried the Helioviewer API [2], making three requests per event to obtain three images images from 72 to 24 hours prior to each event, each at a distance of 24 hours from the others. This interval was chosen because magnetograms taken at shorter intervals tend to exhibit minimal variation, reducing the distinctiveness of temporal features. To ensure uniqueness, we verified that all three images were distinct, as the API may return duplicate images when specific dates are unavailable, this is because it returns the closest available image to the specified date and time. Then we also removed duplicated images that where present in different flare groups leaving a dataset of distinct images for each event. We also checked that the closest available time returned by the API was at a maximum distance of six hours from the requested one in order to not get images that were too different for some samples. When the returned images are outside the six hours range or the returned images are not unique we discarded the whole sample. The images are downloaded and resized to a resolution of 360x360px.

After further examination, HMI images exhibited more background compared to MDI images, and some images had partial visibility or distorted scaling of the Sun. To address these issues, we first cropped rows and columns around the Sun's surface with intensities below a threshold of 10 (considered dark background pixels), then resized the results to 224x224 pixels. It is important to notice that after the cropping we have images of the sun surface that still have different sizes, this is why we needed to reduce the

[1] http://jsoc.stanford.edu
[2] https://api.helioviewer.org/docs/v2/

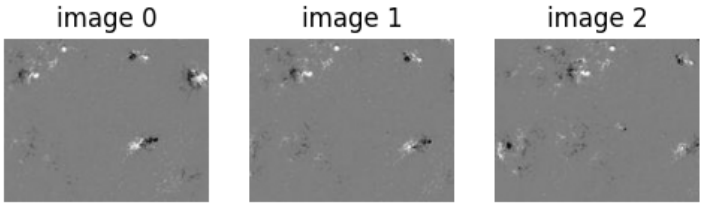

Figure 1: Final images passed to the networks. Image 0 is the sun magnetograms 72 hrs before the event, image 1 is 48 hrs before and image 2 is 24 hrs before.

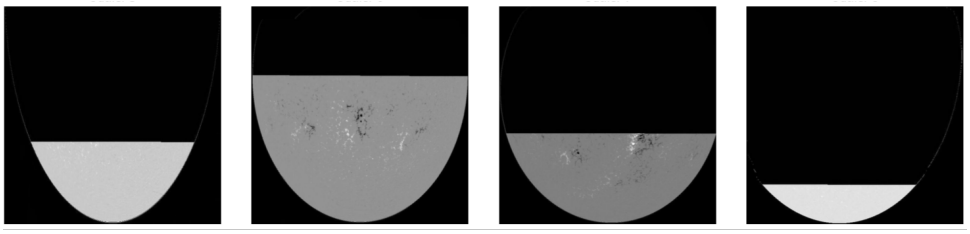

Figure 2: Some outliers found: here you can notice that in some cases the sun surface is partially visible. There are also cases where the image is shrinked or just cutted in different ways. These outliers where identified by studying the distribution of dark pixels in the images.

size to a lower value than the original of 360x360. The final shape of 224x224 was decided after observing that [10] used this size as input for the CNN. After this process we proceeded with outlier identification; the outliers were identified by calculating the percentage of dark pixels (pixel whose intensity values are below 40) in each image, then computing the mean and standard deviation. A z-score of 0.1 was used to identify outliers, this value was determined through visual inspection of the returned outliers. With this procedure, 116 outlier images were identified and removed, see Figure 2 for an example.

Before feeding the data to the various models we also performed a cut on the final resized image in order to get only the central zone of the sun's since the edges of the images can sometimes contain artifacts or other distortions, thus giving the CNN a clearer view of the most important features. Specifically we kept only a square of $width = 180$ and $height = 130$ centered in the original image, see Figure 1. The final step was to standardize the images, this is useful because it normalizes intensity ranges reducing variability from different instruments or conditions, allowing to get better convergence of the models. We applied a Z-score standardization computed globally across all training images, using only training data. This standardization is executed at runtime since we will use a sliding window K-fold validation on the dataset that will use different folds for training at each stage.

4.2. Dataset building

The aim of this study is to detect if a strong solar flare will happen 24 hrs after the last image in the sequence, thus, after downloading all the needed images, we will assign the data corresponding to classes A, B and C to the class 0 (no strong flare) and the images for classes M and X in the class 1 (strong flare). The images are kept in chronological order for the whole study since we are dealing with time series data and we want to conserve realistic prediction conditions.

Each image is compiled into a h5 file for a better handling of the RAM and disk storage. The final dataset contains 2772 elements for class 0 and 1565 elements for class 1, for a total size of 4337 elements. In our dataset the number of samples for the class 1 is 36% of the total size of the dataset while in the study conducted by Yan et al.([10]) the class 1 was 12% of the full dataset of 4694 samples.

Given the class imbalance, we considered applying data augmentation techniques like in the study of Grim et al. [8] that used rotations of 90, 180 and 270 degrees. But they used images of only the sunspots, while in our case we are using images of the full Sun surface. In our case, conventional augmentation methods are not suitable, as they would produce unrealistic results in this context. For instance, applying rotations to the magnetogram images could lead to inaccuracies in sunspot movement among the three different images, which would not correctly reflect the physical rotational motion of the Sun and would lead the network to learn that images that exibit these kind of artifacts are images of the class 1.

5. Methods

Our methodology draws inspiration from the works of Yan et al.[10] and Sun et al.[6]. We aimed to combine their approaches by leveraging both Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs). Specifically, we developed a model that extracts features from three consecutive LOS magnetogram images. The CNN acts as a feature extractor for each image, producing a vector encoding that is then fed into an LSTM, which performs the final classification task. The objective is to predict the occurrence of a strong solar flare (class M or X) 24 hours after the last image in the sequence.

In this study, we explored three different architectures:

- A baseline CNN-LSTM model, where features are extracted by a standard CNN and processed sequentially by an LSTM.
- A model based on an encoder-decoder CNN, where only the encoder part is kept after the training and combined with the LSTM for classification.

Fold	Training Range	Validation Index	Test Range
1	0–3	4	8–9
2	0–4	5	8–9
3	0–5	6	8–9
4	0–6	7	8–9

Table 1
Time Series K-Fold Validation Splits: the test folds are kept fixed during the iterations.

- A Vision Transformer (ViT) based approach, where the three images are concatenated to form a single input image. The transformer is employed to capture relationships across different image patches and perform feature extraction and classification.

Since this study introduces a new dataset and a unique sequence-based modeling approach, results from existing literature using different datasets (e.g., SHARP features or static AR patches) are not directly comparable. Thus, we focus our evaluation on internal baselines across architectural variants (CNN-LSTM, encoder-LSTM, and ViT) under identical experimental conditions. This allows a fair, controlled comparison of modeling strategies on our specific task, while setting a reference point for future work on similar temporal dependent datasets.

5.1. Time series K-fold validation

In time series, we cannot shuffle the data like in normal K-Fold. So we use a method where training comes before validation, and validation comes before test, to keep the time order.

We use 4 folds. In each fold, we increase the training data and move the validation one step forward. The test set is always the same: the last two time steps. See table 1 for visualizing the fold splitting, and table 5 to see how the data is distributed among the folds. From the fold distribution we can see that the test distribution is fixed across folds: 43.5% for class 0 vs 56.5% for class 1, this is good for comparative evaluation since it provides a stable reference point across folds. Folds 1 and 4 have extremely skewed validation sets with 6.7% and 4.2% for class 1, respectively, this could lead to biased model selection during training since we will be using an early stopping criteria based on the validation results.

This method helps to simulate real conditions, where we always train on the past and validate on the future. It is applied after we find the best performing models through some experiments, this is because we wanted to save computational load since we worked under the limits of the GPU usage of the free google colab plan. See next sections for details on each model tested.

The work of Yan et al. [10] used k-folds as well but in their case the k-fold implementation was a standard k-fold with 10 folds.

5.2. Reproducibility

To ensure reproducibility and consistency for all experiments, a fixed configuration was adopted in the entire study. A global random seed (`SEED = 19`) was set and applied for all relevant libraries, including Python's built-in random module, NumPy, and TensorFlow. Deterministic operations were enabled in TensorFlow by setting `TF_DETERMINISTIC_OPS` and `TF_CUDNN_DETERMINISTIC` environment variables, which helps reduce non-determinism of the GPU operations. All models were implemented using TensorFlow and Keras, with standard scientific libraries such as NumPy, Matplotlib, and OpenCV used for data processing and visualization.

5.3. CNN-LSTM

The CNN architecture is inspired by the work proposed by Yan et al. [10]. We first started by testing some models in order to find the best performing one. Then we performed the K-fold validation on this final model. The first training stage is done by dividing the dataset in the train, validation and test with training = 60%, validation = 20% and test = 20% of the dataset size. The training uses Adam optimizer with learning rate of 10^{-3} and an early stopping on the validation loss with patience of 3 used to prevent overfitting.

5.3.1. CNN-LSTM A

The architecture used in this first experiment combines a convolutional neural network (CNN) with a recurrent LSTM layer to handle sequences of image inputs. The base CNN processes each individual image in the sequence and consists of four convolutional blocks. Each block begins with a convolutional layer using ReLU-like activation, followed by batch normalization and max pooling. The first two convolutional layers use 5×5 kernels with 8 and 16 filters respectively, while the following two use 3×3 kernels with 32 and 64 filters. All convolutional layers apply L2 regularization to avoid overfitting and use the Glorot uniform initializer like in the work of Yan et al.[10] to maintain a balanced distribution of weights. After the convolutional feature extraction, a global average pooling (GAP) layer reduces the spatial dimensions, and the resulting feature vector is passed through a dense layer followed by batch normalization. This CNN model is then wrapped in a `TimeDistributed` layer so that it can be applied to each time step in a sequence of images. The outputs from each time step are vectors of dimension 128 that are then fed into an LSTM layer with 128 units, allowing the model to capture temporal dependencies across frames. After the LSTM, the sequence representation is passed through two fully connected layers with 32 and 16 units, both using ReLU activation and dropout of 0.5 for regularization. Finally, the model outputs class probabilities through a softmax layer with 2 neurons.

The use of Global average pooling was employed instead of flattening to reduce the spatial dimensions before the fully connected layers. This choice simplifies the model by significantly reducing the number of parameters. Our experiments indicated that the performance difference between using GAP and flattening were very close, making GAP a more efficient alternative.

5.3.2. CNN-LSTM B

This network is similar to the CNN-LSTM A but it has one extra convolutional layer, before the global average pooling, with 128 filters of kernels 3×3. It is useful to test this second model because adding an extra convolutional layer with more filters can increase the network's capacity to learn deeper and more complex features from the input images. Although CNN-LSTM B adds an extra convolutional layer to increase the model's representational power, the results turned out to be worse compared to CNN-LSTM A. This performance drop can be related to the limited size of the dataset. When a dataset is small, deeper and more complex models are more led to overfitting, meaning they learn specific patterns or noise from the training data that do not generalize well to new, unseen samples. The added convolutional layer introduces more parameters, which require more data to train effectively. Without enough data, the model struggles to learn meaningful features and may perform worse on validation and test sets, despite its higher capacity.

5.3.3. CNN-LSTM C

This third model is like the CNN-LSTM A but the first two layers have kernels of size 11×11. This model was tested because using larger kernels (11×11) in the first two convolutional layers allows the network to capture more global spatial information early in the feature extraction process. Larger kernels are able to cover a wider area of the image, which can be beneficial when important patterns or structures are spread out or not very localized. Despite that, the model performed worse.

5.3.4. CNN-LSTM D

This architecture is similar to CNN-LSTM A but includes an additional LSTM layer with 64 units placed after the original LSTM layer of 128 units. The motivation for testing this variant is to increase the model's capacity to learn more complex temporal patterns by stacking recurrent layers. The results showed a slight lower result compared to the base model, indicating that the extra recurrent layer is not really useful.

5.3.5. CNN-LSTM E

This model follows the same structure as CNN-LSTM A but replaces the smaller fully connected section after the LSTM with a deeper multilayer perceptron (MLP). Specifically, it includes four fully connected layers with 64, 32, 16, and 2 units respectively, each followed by a dropout layer with a rate of 0.5. The intention behind this modification was to allow the network to perform more complex nonlinear transformations after the temporal features are extracted by the LSTM. This architecture produced worse results. The cause is again the small size of the dataset: deeper fully connected networks introduce a large number of parameters, which increases the risk of overfitting. Although dropout was used to reduce this effect, the model may still have learned features that do not generalize well to unseen data.

See a summary of the metrics on the test set for the various architectures on table 2.

See image 3 for the CNN A and LSTM A architecture.

5.4. Encoder-LSTM

The encoder-LSTM (E-LSTM) architecture is based on the same structure described in Section 5.3.1, which was the best model, but in this case the CNN block is pre-trained using an encoder-decoder architecture that is trained on the task of reconstructing the original image passed as input. This pre-training is useful because it helps the model learn a better internal representation of the input data before passing it to the LSTM.

The pre-training stage for the encoder-decoder is performed using an early stopping on the validation loss with patience of 3, the optimizer is Adam with mean square error loss. The bottleneck is again a vector of size 128, the very last layer of the decoder uses a sigmoid activation and during the encoding and decoding step the padding is kept the same.

For this model, we did not run any tests before the k-fold validation because the goal is to investigate how the pre-trained CNN affects the final results.

The k-fold validation is performed using two different configurations. In the first configuration, the CNN weights are kept frozen to preserve the representations learned during the pre-training phase. In the second configuration, the CNN is fine-tuned during training, allowing it to adapt its features to better support the LSTM and possibly improve overall performance. For each stage of the k-fold we will firstly pre-train the CNN in the encoder-decoder and then fine-tune the encoder CNN with the LSTM. The LSTM used is the same of the CNN-LSTM A model.

5.5. Vision Transformer (ViT) Architecture

In this section, we explore the use of a Vision Transformer (ViT) for sequence classification. The ViT model is based on the Transformer architecture (Vaswani et al. [16]), which has shown state of the art results in natural language processing tasks and has been adapted to image processing. The idea is to treat image patches as tokens and apply a Transformer model to capture the long-range dependencies between them.

The input to the Vision Transformer consists of a sequence of three concatenated images, where each image is reshaped to have a height of 130 and a width of 180×3 (since three images are concatenated side by side). The resulting input shape is $(128, 540, 1)$.

Even though the images are compacted into one, the transformer architecture remains capable of learning the temporal relationships between the images due to the following reasons: first, the ViT introduces position embeddings, which encode the relative position of each patch in the sequence. These embeddings allow the model to differentiate between patches from different part of the images in the sequence and understand their correlation. Second, the self-attention layers in the ViT enable the model to capture long-range dependencies between patches, even when they are spatially far away in the concatenated image. This mechanism allows the model to learn which patches (from different images) are relevant to each other in the context of the sequence.

This method of concatenating the images was developed independently and, to the best of our knowledge, has not been previously proposed in related literature about solar flare forecasting.

Among the three described architectures the transformer

Model	Accuracy	Precision	Recall	F1 Score	FAR
CNN-LSTM A	**0.7043**	0.7029	**0.7043**	**0.7027**	**0.2957**
CNN-LSTM B	0.6382	0.6453	0.6382	0.6394	0.3618
CNN-LSTM C	0.6827	0.6816	0.6827	0.6819	0.3173
CNN-LSTM D	0.6971	0.6956	0.6971	0.6949	0.3029
CNN-LSTM E	0.6478	**0.7114**	0.6478	0.5938	0.3522

Table 2
Comparison of CNN-LSTM model performances on the test set. The CNN-LSTM A is the model with the best overall performance.

is the one which suffered most of overfitting, this is due to the large number of parameters of the transformer architecture that led to overfitting on the small dataset. Another reason for the lower result is the missing of inductive bias that is present in the CNN architecture, in particular the ViT does not assume that closer regions of the space are correlated.

Again we trained using a division of the dataset into 60% training, 20% validation and 20% testing, the training is performed with an early stopping on the validation loss and patience of 3.

The final results obtained by the different tested architectures are summarized in table 5.5.6.

5.5.1. ViT A

In this configuration, the model starts by splitting each input image into non-overlapping patches of size 16×16, this patch dimension was chosen according to the famous paper "An image is worth 16x16 words" [17] that demonstrated that parches of 16×16 pixels offer the best threshold between computational cost and effectiveness. Each patch is projected into a vector of dimension 128. The sequence of patch embeddings is then passed through 4 transformer layers, each using 4 attention heads. The feed-forward network inside each transformer block has a hidden dimension of 128. This setup represents a compact transformer architecture, chosen as a starting point to evaluate the performance of ViTs on our dataset while keeping the model size relatively small and suitable for training on our limited data. The model performs well compared to the CNN-LSTM, achieving an accuracy of 0.7068 and an F1 score of 0.6975. These results indicate that even a relatively small transformer can achieve solid performance on this task, setting a reference point for deeper or wider configurations.

5.5.2. ViT B

ViT B increases the projection dimension to 256 while keeping all other parameters equal to those of ViT A. The idea behind this change is to assess whether a higher-dimensional token representation improves the model's ability to learn discriminative features. However, the results slightly declined, with accuracy dropping to 0.6884 and F1 score to 0.6877. This suggests that a higher projection dimension alone, without adjusting other parts of the network, leads to overfitting and inefficient use of model capacity at this scale.

5.5.3. ViT C

In ViT C, we retain the original 128-dimensional projection but increase the MLP dimension from 128 to 256, aiming to evaluate whether a more expressive feed-forward component enhances learning. Compared to ViT A, the model shows a slight performance drop just like ViT B (accuracy of 0.6827, F1 score of 0.6777), indicating that simply enlarging the MLP does not guarantee improved performance and may introduce unnecessary complexity when not supported by sufficient representational capacity in the transformer encoder.

5.5.4. ViT D

ViT D combines the increased projection dimension of ViT B with the larger MLP dimension of ViT C, forming a wider network on both ends. With this architecture, the model explores whether expanding both the embedding and MLP capacity improves overall results. However, the performance continues to decline (accuracy of 0.6564, F1 score of 0.6578), which suggests that increasing model width without modifying the depth or attention capacity leads to diminishing returns, possibly due to optimization difficulties or too many parameters.

5.5.5. ViT E

ViT E deepens the model by using 8 transformer layers and 8 attention heads, in addition to maintaining a projection and MLP dimension of 256. The hypothesis here is that increasing depth and attention complexity might compensate for the lower performances seen in wider but shallow models. The results support this, with accuracy improving to 0.6678 and a higher precision of 0.6956. While it doesn't outperform the baseline, the improvement over ViT D indicates that depth and multi-head attention are more beneficial than width alone in this context.

5.5.6. ViT F

ViT F keeps the deep structure (8 transformer layers and 8 heads) but reverts to the lighter configuration of ViT A in terms of projection and MLP dimensions (128). This is intended to evaluate whether depth alone, without high-dimensional features, can sustain performance. However, the model suffers a substantial performance drop, with an accuracy of 0.5716 and F1 score of 0.5459. This suggests that a deep model without sufficient representational capacity struggles to learn meaningful patterns.

These experiments suggest that model depth contributes more effectively than width to performance improvements in this context, but both are limited by the small dataset size and lack of inductive bias typical of ViTs.

5.6. Data Preprocessing

To efficiently handle the large dataset stored in HDF5 format, we implemented custom data generators to work with the specific structure of the dataset. This approach eliminates the need to store large amounts of data in memory at

Model	Accuracy	Precision	Recall	F1	FAR
ViT A	**0.7068**	**0.7099**	**0.7068**	**0.6975**	**0.2932**
ViT B	0.6884	0.6873	0.6884	0.6877	0.3116
ViT C	0.6827	0.6805	0.6827	0.6777	0.3173
ViT D	0.6564	0.6654	0.6564	0.6578	0.3436
ViT E	0.6678	0.6956	0.6678	0.6670	0.3322
ViT F	0.5716	0.6554	0.5716	0.5459	0.4284

Table 3
Performance metrics comparison for all ViT architectures. The ViT A is the model with best performances.

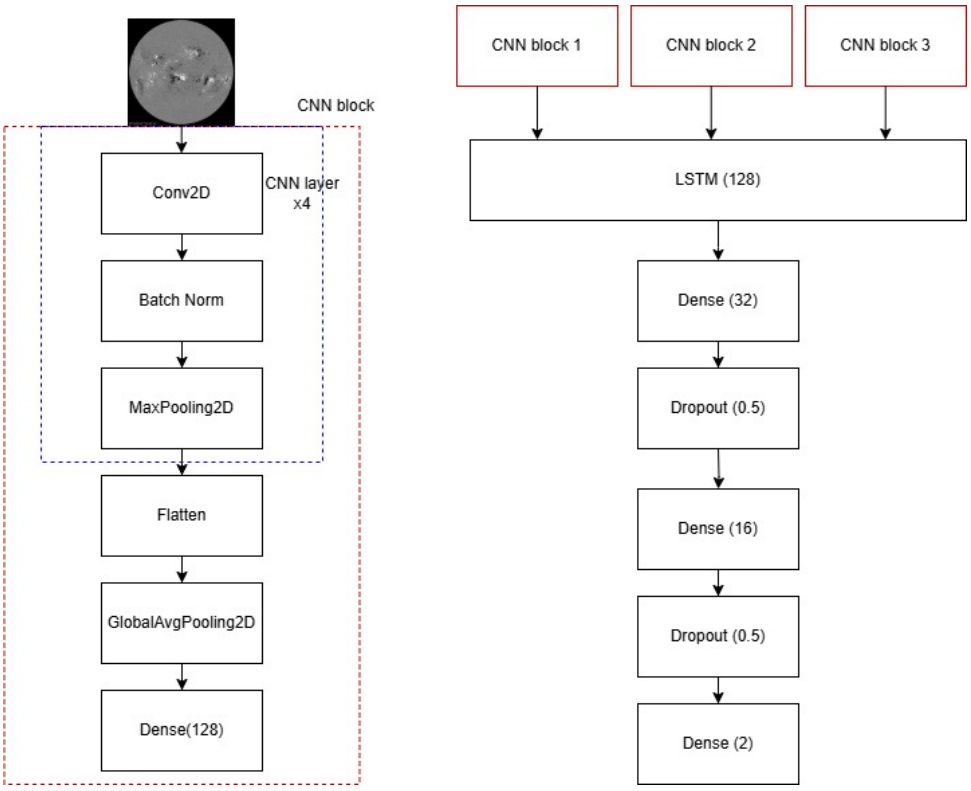

Figure 3: CNN-LSTM architecture: each image from the time series of 72h, 48h and 24h is processed in parallel in the CNN block that will perform the feature extraction making a vector of dimension 128, then the 3 vectors are passed to the LSTM architecture for the final classification stage.

once, enabling us to train on large datasets without running into memory limitations. The generator reads the data in batches, applies necessary preprocessing steps (e.g., resizing, normalization), and yields batches of data of 64 elements with corresponding labels and sample weights for training.

The class balancing is performed by computing sample weights for each class, ensuring that the training process accounts for the imbalanced class distribution.

5.7. Metrics

The metrics we decided to implement are the same already used by the related literature like: [7], [8], [9], [10]. The evaluation metrics used include **Accuracy**, which measures the proportion of correctly classified samples, ranging from 0 to 1, where 1 indicates perfect classification. **Precision** refers to the proportion of correctly predicted positive instances among all predicted positives, also ranging from 0 to 1. **Recall (Sensitivity)** assesses the proportion of actual positive instances that were correctly predicted, with a range of 0 to 1. The **F1-Score** represents the harmonic mean of Precision and Recall, effectively balancing the trade-off between them, and similarly ranges from 0 to 1. Finally, the **False Alarm Ratio (FAR)** measures the proportion of false positives among all positive predictions, ranging from 0 to 1, where 0 means no false positives.

The definition of the above metrics can be seen in table 4.

6. Results

In this section we discuss the results obtained with the best models for the CNN-LSTM, ViT, and the encoder-LSTM architectures. Each model was trained using the K-fold strategy described in Section 5.1, with early stopping on the validation loss and a patience of 3 epochs. Since the dataset is highly unbalanced, we used the Adam optimizer with **categorical cross-entropy** and applied **sample weights** to handle the skewed class distribution during training. Comparative results with other studies are not included, as the dataset used is self-developed. The results we are going to describe are showed in Table 7 and 6.

Metric	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1-Score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}}$
False Alarm Ratio (FAR)	$\frac{FP}{FP+TP}$

Table 4
Evaluation metrics and their corresponding formulas.

Fold	Set	Class 0	Class 1
1	Train	1001 (57.8%)	731 (42.2%)
	Val	404 (93.3%)	29 (6.7%)
	Test	380 (43.5%)	493 (56.5%)
2	Train	1405 (64.9%)	760 (35.1%)
	Val	263 (60.7%)	170 (39.3%)
	Test	380 (43.5%)	493 (56.5%)
3	Train	1668 (64.2%)	930 (35.8%)
	Val	309 (71.4%)	124 (28.6%)
	Test	380 (43.5%)	493 (56.5%)
4	Train	1977 (65.2%)	1054 (34.8%)
	Val	415 (95.8%)	18 (4.2%)
	Test	380 (43.5%)	493 (56.5%)

Table 5
Distribution of the classes in the k fold: while the test set remains constant and balanced across all folds, the validation sets in some folds are highly imbalanced. This imbalance in the validation set can skew model selection, favoring configurations that perform well on the majority class and masking poor performance on the minority class.

6.1. Cross-model comparison

The three architectures results demonstrate different performance profiles across folds. Examining the best-case performance for each model reveals their respective potentials: the Encoder-LSTM frozen achieves the highest single-fold accuracy (0.72 on Fold 2), followed closely by the ViT (0.70 on Fold 3) and Encoder-LSTM unfrozen (0.71 on Fold 2), while the CNN-LSTM reaches 0.68 on Fold 4.

The relationship between training set size and performance varies significantly between architectures. Fold 4, which contains the largest training set (3031 samples), shows this divergence. The CNN-LSTM achieves its best performance on this fold (0.68 accuracy), showing a clear benefit from the additional training samples. The Encoder-LSTM unfrozen also performs well (0.70 accuracy), demonstrating that the architecture can hold larger datasets when the encoder is allowed to adapt. In contrast, the ViT achieves its worst performance on Fold 4 (0.63 accuracy), and the Encoder-LSTM frozen gets the lowest (0.56 accuracy, 0.32 precision). The common factor explaining these failures is Fold 4's severely imbalanced validation set (95.8% class 0, only 18 samples of class 1). The validation loss decreases as models learn to predict the majority class or fails to detect when fixed representations are not good (causing low results in Encoder-LSTM frozen). The low number of minority class samples (only 18) means validation metrics become noisy and unreliable for assessing generalization to the more balanced test set. More balanced validation sets in Folds 2 and 3 (60.7% and 71.4% class 0) produce better results for most models. The ViT achieves its best performance on Fold 3 (0.70 accuracy), while both Encoder-LSTM variants achieve their best results on Fold 2 (0.71 and 0.72 accuracy). Fold 1 presents a different challenge. It has a severely imbalanced validation set (93.3% class 0) and the smallest training set (1732 samples). This combination produces the worst results for CNN-LSTM (0.55 accuracy) and some of the lowest for other architectures.

The confusion matrices in Table 6 reveal that validation set imbalance influences the class prediction biases learned. This provides deeper insight into the performance variations across folds. The CNN-LSTM shows a clear bias towards class ABC in folds 1 and 2 where the model has few training samples and also imbalanced validation sets (93% and 60%). The ViT shows a bias towards the class MX in the first three folds and towards the class ABC in the last one. The encoder-LSTM unfrozen shows a bias towards the MX class in the first and last folds, and a bias towards the ABC class in the third one, while the frozen has always a bias towards the MX class with a complete failure on Fold 4.

Given the limited number of folds (K=4), small differences in mean performance (e.g., 0.66 vs 0.65 accuracy) are not statistically significant. Hovewer, the ViT and Encoder-LSTM unfrozen show similar average performance with slightly lower variance than the frozen variant, while the CNN-LSTM is the most stable under adverse validation conditions. Across all models, the dominant factor affecting performance is validation imbalance rather than architectural differences.

6.2. CNN-LSTM A

The CNN-LSTM demonstrates progressive improvement in performance as training data increases. The model achieves its best results on Fold 4 with accuracy of 0.68, precision of 0.69, F1 of 0.68, and FAR of 0.32. This represents an high improvement over Fold 1 (0.55 accuracy and F1), which contains the smallest training set (1732 samples). The correlation between training set size and performance is clear: Fold 4 (3031 samples, 0.68 accuracy) > Fold 3 (2598 samples,

Table 6
Confusion Matrices per Fold for Each Model (Classes: ABC (0), MX (1))

Model	Fold	True / Pred	ABC	MX
CNN-LSTM	Fold 1	ABC	**244**	136
		MX	**259**	234
	Fold 2	ABC	**333**	47
		MX	**337**	156
	Fold 3	ABC	**269**	111
		MX	186	**307**
	Fold 4	ABC	**278**	102
		MX	175	**318**
ViT	Fold 1	ABC	102	**278**
		MX	27	**466**
	Fold 2	ABC	92	**288**
		MX	9	**484**
	Fold 3	ABC	182	**198**
		MX	65	**428**
	Fold 4	ABC	**304**	76
		MX	**249**	244
Encoder-LSTM Unfrozen	Fold 1	ABC	89	**291**
		MX	12	**481**
	Fold 2	ABC	**244**	136
		MX	120	**373**
	Fold 3	ABC	**295**	85
		MX	**308**	185
	Fold 4	ABC	140	**240**
		MX	20	**473**
Encoder-LSTM Frozen	Fold 1	ABC	57	**323**
		MX	5	**488**
	Fold 2	ABC	171	**209**
		MX	39	**454**
	Fold 3	ABC	188	**192**
		MX	61	**432**
	Fold 4	ABC	0	**380**
		MX	0	**493**

0.66 accuracy) > Fold 2 (2165 samples, 0.56 accuracy) > Fold 1 (1732 samples, 0.55 accuracy).

The confusion matrices (Table 6) reveal how the CNN-LSTM's prediction strategy changes across folds. Fold 2 shows an extreme bias toward ABC (Class 0), with 670 ABC predictions versus only 203 MX predictions (77% ABC prediction rate). This explains the fold's poor performance (0.56 accuracy) despite having a good amount of training data and a balanced validation set, the model learned an inappropriate class prediction that does not match the test distribution. In contrast, Folds 3 and 4 show more balanced predictions (418 and 420 MX predictions out of 873 samples, or 48%). This improved balance corresponds directly to the improved accuracy on these folds (0.66 and 0.68).

This consistent benefit from additional training data, combined with increasingly balanced class predictions as training data grows, even though the validation set distribution is more and more skewed, distinguishes the CNN-LSTM from the higher-capacity ViT architecture and the encoder-LSTM.

The performance on Fold 1, despite its relatively balanced training set (57.8% class 0), reveals that insufficient training data is a fundamental limitation for training from scratch. With only 1732 training samples, the model cannot learn a good representations regardless of validation set characteristics. In contrast, the Encoder-LSTM benefits from pre-trained weights that provide better initialization even with limited task-specific training data.

In summary, the CNN-LSTM provides a solid baseline. Its consistent improvement with data scale and stable behavior across validation conditions make it a reliable choice for this task.

6.3. ViT A

The ViT A reaches its best performance on Fold 3 (accuracy 0.70, precision 0.71, F1 0.68, FAR 0.30). Performance are very different across folds (0.63–0.70 accuracy), with differences explained by validation set composition and resulting class prediction biases. Among all three architectures, the ViT shows the most extreme and inconsistent prediction

CNN-LSTM A						
	Fold	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**FAR**
	1	0.5475	0.5683	0.5475	0.5468	0.4525
	2	0.5601	0.6503	0.5601	0.5292	0.4399
	3	0.6598	0.6721	0.6598	0.6611	0.3402
	4	**0.6827**	**0.6947**	**0.6827**	**0.6839**	**0.3173**
	mean ± std	0.61 ± 0.07	0.65 ± 0.06	0.61 ± 0.07	0.61 ± 0.08	0.39 ± 0.07
ViT A						
	Fold	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**FAR**
	1	0.6506	0.6979	0.6506	0.5999	0.3494
	2	0.6598	**0.7506**	0.6598	0.5986	0.3402
	3	**0.6987**	0.7068	**0.6987**	**0.6847**	**0.3013**
	4	0.6277	0.6699	0.6277	0.6226	0.3723
	mean ± std	0.66 ± 0.03	0.71 ± 0.03	0.66 ± 0.03	0.63 ± 0.04	0.34 ± 0.03
Encoder-LSTM						
Unfrozen	**Fold**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**FAR**
	1	0.6529	0.7354	0.6529	0.5905	0.3471
	2	**0.7068**	0.7056	**0.7068**	**0.7059**	**0.2932**
	3	0.5498	0.5999	0.5498	0.5351	0.4502
	4	0.7022	**0.7555**	0.7022	0.6687	0.2978
	mean ± std	0.65 ± 0.07	0.70 ± 0.07	0.65 ± 0.07	0.63 ± 0.08	0.35 ± 0.07
Frozen	**Fold**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**FAR**
	1	0.6243	0.7400	0.6243	0.5349	0.3757
	2	**0.7159**	**0.7411**	**0.7159**	0.6959	**0.2841**
	3	0.7102	0.7196	0.7102	**0.6970**	0.2898
	4	0.5647	0.3189	0.5647	0.4076	0.4353
	mean ± std	0.65 ± 0.07	0.63 ± 0.21	0.65 ± 0.07	0.58 ± 0.14	0.35 ± 0.07

Table 7
K-fold evaluation results for CNN-LSTM A, ViT A, and Encoder-LSTM (frozen and unfrozen). Bold values indicate the best performance per model configuration.

biases across folds. When validation sets were highly imbalanced (Folds 1 and 4), the model developed strong class biases, either predicting mostly MX or mostly ABC. This indicates that the ViT's large capacity makes it more sensitive to misleading validation feedback, even with class weighting applied. When the validation set was moderately balanced (Fold 3), the model generalized best, suggesting that the ViT can perform competitively but requires reliable validation signals. The confusion matrices confirm that the ViT's bias direction and magnitude change drastically with validation set composition, making it the least stable architecture despite having competitive mean performance (0.66 ± 0.03 accuracy). Overall, the ViT A can achieve competitive performance when class weighting is combined with reasonably balanced validation sets (Fold 3), but it shows the most extreme and unpredictable biases when validation feedback is poor in imbalanced conditions. For deployment, this architecture requires careful validation set curation or alternative stopping criteria to prevent learning inappropriate biases.

6.4. Encoder-LSTM

The Encoder-LSTM was evaluated with frozen and unfrozen encoder weights. Both reach similar mean accuracy (0.65 ± 0.07) but show very different behaviors across folds.

6.4.1. Unfrozen encoder

Allowing the encoder to adapt improves stability and performance. The best results appear in Fold 2 (accuracy 0.71, precision 0.71, F1 0.71, FAR 0.29), where confusion matrices show balanced predictions between ABC and MX. Fine-tuning helps the model exploit larger training sets and correct encoder data mismatches. Interestingly, it fails on Fold 3 (accuracy 0.55, FAR 0.45), showing strong ABC bias (69% predictions). Here, fine-tuning likely overfits to the ABC skewed validation set (71% class 0), pulling the encoder away from useful pre-trained features.

6.4.2. Frozen encoder

The frozen version can perform well on some folds (up to 0.72 accuracy in Fold 2) but is extremely unstable. It shows persistent MX bias, predicting MX for 70–95% of samples, and collapses completely on Fold 4 by predicting MX for every case (accuracy 0.56 only because MX dominates the test set). The failure is driven by two factors: fixed encoder features that do not generalize to the new split, and again the highly imbalanced validation set (95.8% class 0) that misleads early stopping.

The frozen model achieves higher peak accuracy but poor reliability, while the unfrozen model generalizes better in general, but remains sensitive to validation imbalance. Compared to ViT, both Encoder-LSTM variants show stronger bias under skewed validation signals. These results suggest that the pre-trained encoder captures only partially relevant representations, and that fine-tuning helps only when the validation feedback is balanced enough to guide adaptation.

7. Explainability Analysis

To understand which spatial regions the models are focusing on when making predictions, we applied Gradient-

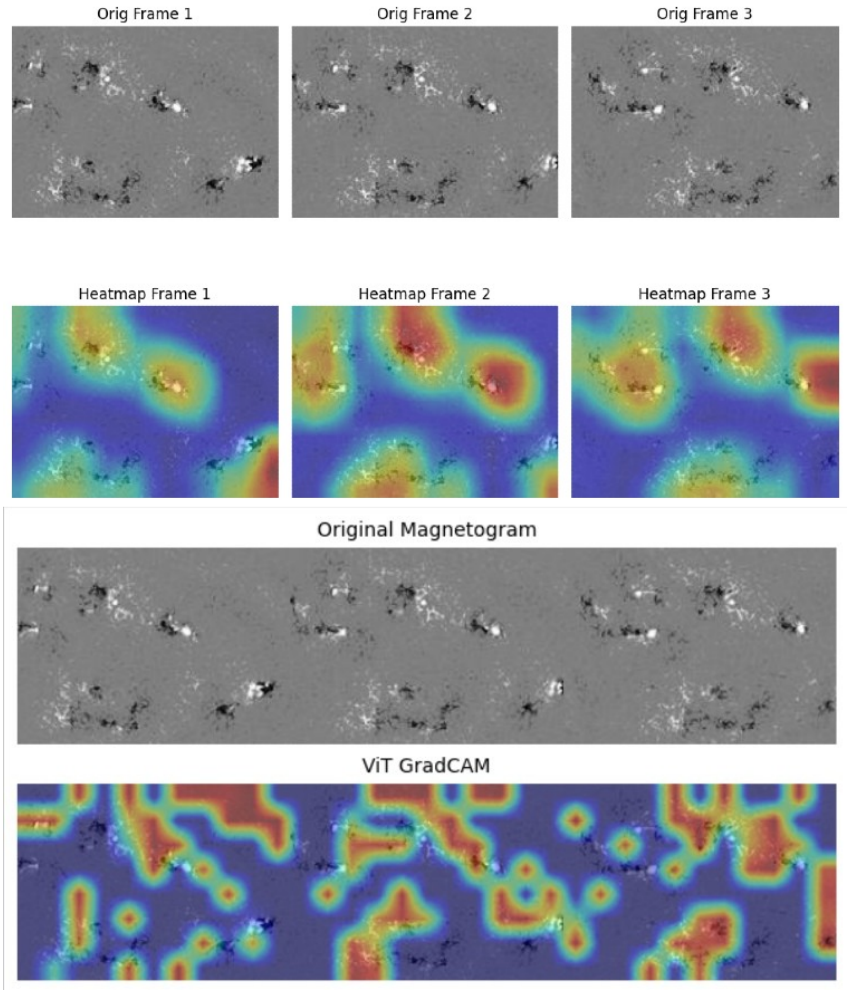

Figure 4: Grad-CAM visualizations for a correctly predicted solar flare event (Class 1) from the test set. **Top:** CNN-LSTM attention across three temporal frames shows focused, evolving attention on specific active regions (bright areas in the original magnetograms). The model tracks the same regions across time, with attention intensifying as the sequence progresses. **Bottom:** ViT attention (using layer normalization 3) shows more distributed attention across multiple regions, reflecting the transformer's global processing strategy. Red indicates high attention, blue indicates low attention. Both models correctly predicted the flare.

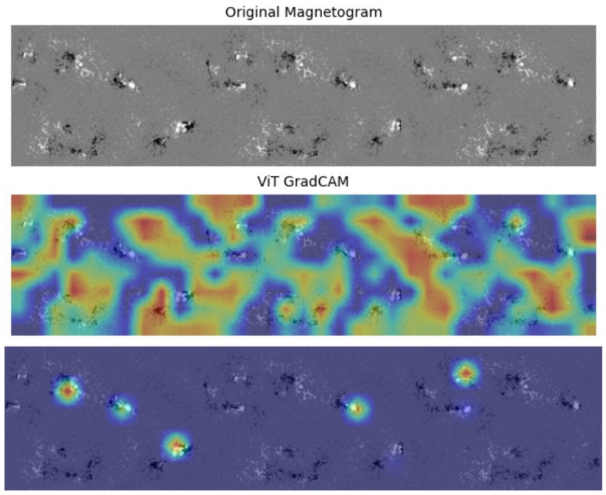

Figure 5: Grad-CAM visualizations for the ViT layer 1 (top) and layer 8 (bottom).**Top:** Layer 1 shows diffuse, global attention as the model establishes context. **Bottom:** Layer 8 (final layer) shows highly concentrated attention but may be overly specific to training patterns. Layer 3 was selected for the main analysis in figure 4 (bottom) as it balances interpretability with representativeness of the model's attention strategy.

weighted Class Activation Mapping (Grad-CAM) [18] to the CNN-LSTM and ViT architectures. Grad-CAM generates heatmaps that highlight the input regions most important for the model's classification decision, allowing us to visualize if the models are paying attention to physically relevant solar features.

The Encoder-LSTM was not included in this analysis because it shares the same convolutional architecture as the CNN-LSTM, since both models use identical CNN structures for spatial feature extraction, the Grad-CAM visualizations have highlighted similar spatial regions to the CNN-LSTM in both its configurations. The primary difference between these models lies in their weight initialization rather than in the fundamental spatial attention patterns that Grad-CAM reveals.

The sample chosen for this test is a sample of an element of class 1 (Flare) from the test set after training the models with the division train-val-test (60%-20%-20%), in this case both models correctly identified the class.

The original magnetograms was chosen because it has multiple active regions characterized by bright (strong positive magnetic field) and dark (strong negative magnetic field) areas, which are typical indicators of solar activity.

The CNN-LSTM heatmaps across the three temporal frames shows an evolving attention pattern. In Frame 1, the model focuses mainly on two distinct active regions: a bright region in the upper portion of the magnetogram and another in the lower-right. As the sequence progresses to Frame 2, attention intensifies on the upper regions. In Frame 3, the attention pattern shifts again, continuing to track the same active regions as they evolve. This temporal progression demonstrates that the CNN-LSTM has learned to track specific active regions across the sequence. The localized spot of the attention, focused on dark and bright regions, suggests the model has correctly learned what is important in the image.

The ViT heatmap presents a different attention strategy. Rather than focusing on discrete regions, the ViT shows distributed attention on multiple areas of the magnetogram. While several "hot spots" (yellow-red areas) align with bright active regions visible in the original image, the attention is more diffuse than the CNN-LSTM, with moderate activation (green-yellow) spread across much of the solar disk.

The choice of layer normalization layer for ViT visualization affects the apparent attention distribution, see image 5 for details. Deeper layers show more refined attention, but this analysis used layer 3 to capture the model's intermediate attention strategy. This layer dependence itself reveals the hierarchical nature of transformer processing: early layers attend to establish global context, while deeper layers refine attention based on that context.

This analysis presents a single example from the test set. While this example demonstrates the distinct attention strategies of the two architectures, a comprehensive explainability study would require analysis of multiple samples including true negatives, false positives, and false negatives to understand when and why the models fail.

8. Discussion

The results presented in Section 6 highlight several important findings regarding the behavior of the CNN-LSTM, ViT, and Encoder-LSTM architectures under varying data distributions and training conditions.

The model performance is strongly influenced by the class imbalance present in the validation sets used for early stopping. While the K-fold approach provides an estimate of model generalization, the high imbalance in some folds introduces a bias in the model selection. This shows that validation signal is more critical than raw dataset size for the final model performance. This is also proven by the confusion matrices, that show that all models can develop strong and opposite biases across folds. The ViT alternates between MX-dominated and ABC-dominated predictions depending on the validation composition, while the Encoder-LSTM shows similar instability with less extreme swings.

Pre-training still offers an advantage. The unfrozen Encoder-LSTM performs better than the CNN-LSTM on folds with moderate balance and low samples, confirming that pretrained representations help when training data are limited. But this benefit depends on allowing the encoder to adapt, since the frozen version often collapses when its fixed features do not match the new data split.

Among the three models, the ViT reaches the best mean performance and most balanced results when the validation set is reasonably representative (Fold 3). However, the same capacity also makes it more unstable under poor validation feedback, meaning that large models need stronger regularization or more reliable stopping criteria.

The Grad-CAM analysis supports these quantitative findings. The CNN-LSTM shows spatially localized and temporally consistent attention, indicating that it tracks evolving active regions over time, this behavior aligns well with domain understanding of solar flare generation. The ViT instead exhibits distributed attention across the solar disk, reflecting its ability to integrate global spatial context.

Finally, it is important to note that differences in mean performance across folds (e.g., 0.66 vs 0.65 accuracy) are not statistically significant given $K = 4$. The trends discussed here should be interpreted as qualitative indicators and not definitive rankings.

9. Conclusion

This work compared three deep learning architectures: CNN-LSTM, Vision Transformer (ViT), and Encoder-LSTM, for solar flare prediction using a self-developed dataset. The study shows that model performance changes drastically with data distribution and validation strategy.

The ViT reached the highest single-fold and mean performance but also showed unstable behavior across folds. The confusion matrices revealed strong and opposite biases depending on the validation composition. The model can generalize well when the validation feedback is balanced, but it easily overfits to the majority class when imbalance is high. This confirms that large-capacity models require careful regularization and reliable stopping criteria.

The Encoder-LSTM, especially when fine-tuned (unfrozen), provided competitive results and showed the benefits of pre-trained spatial encoders for data-efficient learning. However, the frozen version developed strong biases for the class 1 (MX). The CNN-LSTM, while simpler and computationally cheaper, improved steadily with more training data and remained the most stable model when validation

imbalance was high.

Overall, the analysis shows that validation set imbalance can make early stopping based on loss distorted, leading to misleading model selection. Future work should prioritize balanced validation splits or use metrics and stopping criteria more robust to class imbalance.

10. Future Work

Future work will aim to improve both model performance and methodology. The explainability analysis should be expanded by applying Grad-CAM to a larger number of test samples, including false positives and false negatives, to verify whether the attention patterns remain consistent among different cases.

Another important direction is increasing the number of folds in the cross-validation, which would provide a more reliable estimate of model variance and the statistical significance of observed differences.

For practical applications, future work should test the models in real-time conditions to evaluate their speed and reliability when processing continuous streams of solar observations. Extending the temporal input window beyond the current three frames and increasing the sampling cadence, for example to 12-hour intervals, could help the models learn longer-term evolution patterns and capture gradual build-up phases that lead to major solar flares. Using higher-resolution magnetograms would also allow finer spatial details to be represented, improving the models' ability to detect subtle magnetic structures.

The ViT model in this study processed the three temporal frames through simple concatenation. A dedicated temporal ViT architecture should be developed and tested to evaluate whether explicit temporal modeling improves performance and generalization. Finally, future research should focus on integrating multiple data sources in a unified deep learning framework. Combining magnetograms with SHARP summary parameters or other solar activity indices could help capture both localized magnetic field evolution and broader physical conditions. Such multimodal integration may lead to models that can better detect the early signs of strong solar flares that are not visible from magnetograms alone.

References

[1] S. Sinha, O. Gupta, V. Singh, B. Lekshmi, D. Nandy, D. Mitra, S. Chatterjee, S. Bhattacharya, S. Chatterjee, N. Srivastava, et al., A comparative analysis of machine-learning models for solar flare forecasting: identifying high-performing active region flare indicators, The Astrophysical Journal 935 (2022) 45.

[2] H. Isobe, T. Takahashi, D. Seki, Y. Yamashiki, Extreme solar flare as a catastrophic risk, Journal of Disaster Research 17 (2022) 230–236.

[3] P. H. Scherrer, J. Schou, R. Bush, A. Kosovichev, R. Bogart, J. Hoeksema, Y. Liu, T. Duvall, J. Zhao, A. Title, et al., The helioseismic and magnetic imager (hmi) investigation for the solar dynamics observatory (sdo), Solar Physics 275 (2012) 207–227.

[4] J. T. Hoeksema, Y. Liu, K. Hayashi, X. Sun, J. Schou, S. Couvidat, A. Norton, M. Bobra, R. Centeno, K. Leka, et al., The helioseismic and magnetic imager (hmi) vector magnetic field pipeline: overview and performance, Solar Physics 289 (2014) 3483–3530.

[5] P. H. Scherrer, R. S. Bogart, R. Bush, J.-a. Hoeksema, A. Kosovichev, J. Schou, W. Rosenberg, L. Springer, T. Tarbell, A. Title, et al., The solar oscillations investigation—michelson doppler imager, The soho mission (1995) 129–188.

[6] Z. Sun, M. G. Bobra, X. Wang, Y. Wang, H. Sun, T. Gombosi, Y. Chen, A. Hero, Predicting solar flares using cnn and lstm on two solar cycles of active region data, The Astrophysical Journal 931 (2022) 163.

[7] S. Zhang, Y. Zheng, X. Li, H. Ye, L. Dong, X. Huang, P. Yan, X. Li, J. Wei, C. Xiang, et al., A novel solar flare forecast model with deep convolution neural network and one-against-rest approach, Advances in Space Research 74 (2024) 3467–3480.

[8] L. F. L. Grim, A. L. S. Gradvohl, Solar flare forecasting based on magnetogram sequences learning with multiscale vision transformers and data augmentation techniques, Solar Physics 299 (2024) 33.

[9] P. Sun, W. Dai, W. Ding, S. Feng, Y. Cui, B. Liang, Z. Dong, Y. Yang, Solar flare forecast using 3d convolutional neural networks, The Astrophysical Journal 941 (2022) 1.

[10] P. Yan, X. Li, Y. Zheng, L. Dong, S. Yan, S. Zhang, H. Ye, X. Li, Y. Lü, Y. Ling, et al., A real-time solar flare forecasting system with deep learning methods, Astrophysics and Space Science 369 (2024) 110.

[11] E. Park, Y.-J. Moon, S. Shin, K. Yi, D. Lim, H. Lee, G. Shin, Application of the deep convolutional neural network to the forecast of solar flare occurrence using full-disk solar magnetograms, The Astrophysical Journal 869 (2018) 91.

[12] K. Kaneda, Y. Wada, T. Iida, N. Nishizuka, Y. Kubo, K. Sugiura, Flare transformer: Solar flare prediction using magnetograms and sunspot physical features, in: Proceedings of the Asian Conference on Computer Vision (ACCV), 2022, pp. 1488–1503.

[13] H. Liu, C. Liu, J. T. Wang, H. Wang, Predicting solar flares using a long short-term memory network, The Astrophysical Journal 877 (2019) 121.

[14] A. Chen, Q. Ye, J. Wang, Flare index prediction with machine learning algorithms, Solar Physics 296 (2021) 150.

[15] W. T. Barnes, M. G. Bobra, S. D. Christe, N. Freij, L. A. Hayes, J. Ireland, S. Mumford, D. Perez-Suarez, D. F. Ryan, A. Y. Shih, et al., The sunpy project: Open source development and status of the version 1.0 core package, The Astrophysical Journal 890 (2020) 68.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[17] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, arXiv preprint arXiv:2010.11929 (2020).

[18] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 618–626.