Leiden University

Advances in Deep Learning

Solar Activity and Property Predictions using Deep Learning Methods

Authors:
Mariam Abdallah (s2777126)
Peter Breslin (s3228282)
Alberto Brentegani (s2652838)
Matthijs van Groeningen (s2675544)
Max Moynan (s2908700)
Louis Siebenaler (s3211126)

June 24, 2022

Contents

1 Introduction

Within the last decade, the field of solar physics (i.e. the study of the Sun and its activity) has advanced significantly. The number of dedicated solar missions and telescopes has risen during this time and considerable developments have been made to observational instruments as a result. In particular, improvements to instrument temporal, spatial and spectral resolution has driven an exponential increase in the volume of data being collected within this field. Such a rapid influx of high-resolution observations poses an enormous data challenge to researchers, with the ability to store and handle data becoming increasingly difficult. Sorting through this is simply not feasible for humans and automated methods must therefore be employed.

With advancements in neural networks continuing at an accelerating rate, it is not difficult to realize the potential of machine learning applications to astronomical fields. Reducing the dimensionality of data is a prime use-case of such applications, a technique that is becoming increasingly relevant in the age of big data astronomy. Furthermore, the field of deep learning has opened the door to a range of more complex applications beyond the scope of just data handling. Neural networks with deep layers and abstract architectures can drastically improve the performance of certain tasks and can be much better and faster than traditional algorithms. In particular, deep learning tools can excel at the detection of features or patterns within a dataset that classical algorithms fail to notice. Such methods can be applied to numerous area within the field of solar physics.

1.1 Background

Internal mechanisms within the Sun drive complex magnetic interactions throughout its structure. Much of this appears on the surface as intricate motions of plasma, resulting in a complicated entanglement of magnetic fields that change with time. It is this time-dependent variation that defines solar activity. Active regions (ARs), also called sunspots, describe the large-scale magnetic activity across the Sun's surface. These are the source regions for *solar eruptions*, expulsions of radiation and plasma due to the build-up and subsequent release of magnetic energy. Solar flares and coronal mass ejections are examples of these *solar eruptive events*, both of which are attributed with the release of highly-energetic charged particles. This can have a range of consequences on human activities, including interference with both ground- and space-based technological systems and wide scale communication blackouts. Furthermore, astronauts in space beyond the protection of Earth's magnetic field risk severe radiation sickness from these events, a factor that will become increasingly more important as our species looks towards travelling beyond planet Earth.

Space weather refers to the time-varying conditions on the Sun and in the near-Earth environment that can have an influence on human activity (Hanslmeier 2007). Forecasting space weather events can help in predicting the occurrence of solar eruptions and helps to mitigate any adverse effects. However, this is a difficult task to accomplish well. More accurate space weather forecasts can be achieved by developing our understanding of the processes involved in solar eruptions, both leading up to and during each event. Although solar eruptive events are believed to originate from ARs, their formation and eruption mechanisms are rather poorly understood due to the inherently chaotic nature of solar activity. Deep learning presents the opportunity to predict the behavior of

AR properties and detect features and patterns that may otherwise be too difficult for classical automated methods. The application of neural networks within this field therefore provides tools to better understand and forecast solar eruptions.

1.2 Project direction

This project was first inspired by a study by Armstrong & Fletcher (2019) in which deep learning was used for image classification of different solar features. This was done so in part to address the data challenge in solar physics and provide an automated tool to more efficiently detect and classify solar features for later study. As will be detailed in Section 2, their work consisted of a convolutional neural network (CNN) that was trained and evaluated on a dataset compiled by the authors. This study served as a good starting point to begin our project with as there were a number of improvements and suggestions from the authors that could be added, hence we began with the plan to further explore this work. However, the scope of the project had to redefined due to irreparable complications discovered with the dataset from their work, as will be discussed in Section 2. During this time, we also worked with a separate dataset of AR images collected from Fang et al. (2019) with the purpose of exploring different CNN architectures to better classify ARs. Although our results were positive, the task of image classification in this context was not extremely difficult and it was deemed less interesting than other deep learning prospects within this field. Therefore, when it was realized that we could no longer continue with our original plan, we decided to move away from image classification and switch focus to what we saw as a more engaging and functional challenge: solar activity predictions. This was motivated by the importance of space weather forecasting and how deep learning offers the ability to help better predict and understand solar events. Furthermore, there is an abundance of publicly available sources online for observational data related to solar eruptive events.

With this new focus in mind, we decided to work on two forms of prediction efforts. Firstly, we attempted to predict the time-evolution for a range of AR properties. As previously mentioned, eruptive solar activity is believed to originate from ARs. Predicting how their magnetic properties evolve with time may alert researchers of potential eruptions if their values are predicted to cross certain threshold levels. Furthermore, this could also reveal correlations between properties that were previously unknown or poorly understood. Secondly, we attempted to predict if certain ARs would produce a solar flare within a certain time-range based on a number of time-sequenced AR properties. Solar flares are one of the most common and potentially hazardous forms of eruptive solar events, and therefore their occurrence would be advantageous to predict. Flares come in forms of varying intensity, meaning that some can be more impactful towards human activities than others. This can be of great significance for space-bound communication systems, for example, when deciding whether to shut down operations due to the expected arrival of a solar flare that has been predicted to be sufficiently hazardous towards their equipment. Extending our prediction efforts to include a classification of which type of flare may occur would therefore be both an interesting and useful functionality to add, and hence why we also tackled this challenge. Due to the nature of the time-series datasets acquired, we decided to explore both LSTMs and Transformers for these tasks, as will be discussed in Section 3.

2 Armstrong

2.1 Dataset

In this section, we implement the works of Armstrong & Fletcher (2019) and train a CNN to classify solar images based on its type of solar activity. The solar image data was obtained from the Hinode/SOT Hα (6563 Å) instrument within the visible wavelength range. The dataset consists of 131275 gray-scale images with a pixel size of 256×256 that correspond to 166 solar events. The images were manually labeled with five possible solar activity/event classes; filaments, prominences, flare-ribbons, sunspots, and quiet sun, as illustrated in Figure A1, with training images of 1578, 2735, 2420, 2929, and 2195 respectively.

2.2 Model Architecture and Training Scheme

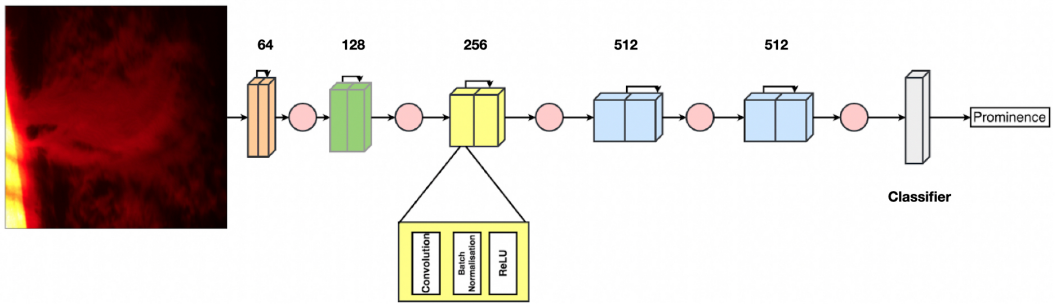

Figure 1: The CNN network architecture used is a VGG13 network that consists of 10 convolutional layers followed by a classifier consisting of 3 dense layers.

The network architecture is illustrated in Figure 1. The architecture is a VGG13 network that consists of 10 convolutional layers, each followed by batch normalization. The convolutional layers are followed by the classifier that consists of 3 dense layers with dropout layers in between to prevent overfitting. The original implementation was in `Pytorch`, however, we have translated the architecture to `Tensorflow` for convenience and instead used separable convolutional layers to obtain a performance boost. The training proceeds with a training-test dataset split of $90\% - 10\%$ (as implemented in the original publication), the *Sparse Categorical Cross Entropy* loss function, the Adam optimizer with a learning rate of $0.5 \cdot 10^{-6}$, and a batch size of 128.

2.3 Result and Investigations

The left panel of Figure 2 shows the results for the trained network. As we can see, the network achieves a very high accuracy of 96.7% on the training set and an even higher accuracy on the validation set of 97.7%, which we found suspicious. As a result, we investigated the data further and discovered two major issues with the data. The first issue is that for each solar event in the data, its images were only taken minutes apart, therefore; all the images in the dataset within each class are highly similar to each other. The second issue is that for each event in the data, 90% of the captured images constitute the training set while the remaining 10% constitute the validation set. And given how similar these images are, we essentially end up with the exact same images

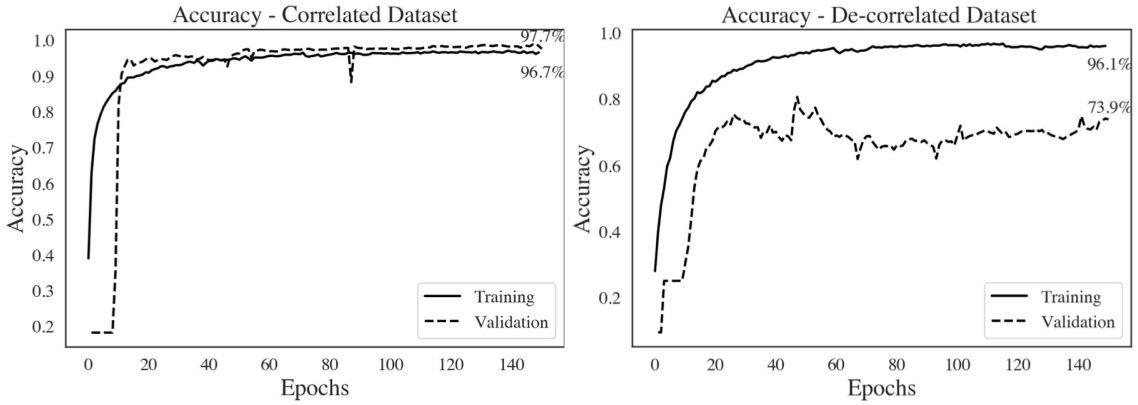

Figure 2: **Left Panel:** network's accuracy trained on the correlated dataset, **Right Panel:** network's accuracy trained on the de-correlated dataset

in the training and validation sets. As a result, we have been dealing with a highly correlated dataset. Our first course of action was to decorrelate the data by ensuring that the solar events in the training set are separate from the solar events in the validation set with no overlap, then we repeated the training process. The results for which are illustrated in the right panel of Figure 2. The accuracy on the training set is still very high, however; the accuracy on the validation set has significantly dropped. This is because the images within the training dataset are not diverse enough to allow the network to learn sufficient feature representations for each class. As a result, the network overfitted the data and simply memorized the training images and thus failed to generalize when applied to the validation set.

In an effort to inject some diversity in the data to salvage it, we have attempted to augment it using various transformation schemes, namely rotation, salt and pepper noise, altering sharpness, flipping, and altering contrast. To ensure that the augmentation will increase the diversity of images, we randomized the number of transformations applied to each image. For example, one image could be flipped with increased contrast, while another could be randomly rotated with noise added and contrast increased. An example of the data augmentation is illustrated in Figure A2. However, the results did not improve even with augmentation leading us to conclude that the data lacks the diversity needed to be usable and is simply unsalvagable. We have informed the authors of the original publication of our results, however; they have not responded yet.

3 LSTMs and Transformers

Recurrent Neural Networks (RNNs) differ from traditional feed-forward networks as the output of each recurrent layer at each time step is used as an additional input for the same layer at the next time step. Therefore, theoretically RNNs should be able to model any sequential data with dependencies between data points, such as those in time series. However, in practice vanilla RNNs struggle to learn dependencies between data points that are more than 10 steps apart. Additionally, RNNs are effectively deep feed-forward networks when you unwrap them. As a result, they often suffer from exploding or vanishing gradients, and have no possibility to parallelize within a series, leading to slow execution speed.

Long Term Short Memory Networks (LSTMs) are a modified type of RNN, known as gated RNNs, which allow them to learn longer term dependencies (Hochreiter & Schmidhuber 1997). They do this by including an additional state, the cell state, whose purpose is to store long-term information. Therefore, the output of each layer is a function of the input, as well as the previous time step's hidden state and cell state. Inside a LSTM cell there are 3 gates which allow the network to learn the longer dependencies. First, the forget gate learns which part of the cell state to remove given the input and the previous time step's hidden state. Then, the input gate learns which portions of the input and previous time step's hidden state need to be added to the cell state. Finally, the output gate learns which part of the cell state to add to the hidden state to form the desired output. Overall, these cells can be used in a variety of different architectures, mainly seq2vec, vec2seq and seq2seq problems. Nevertheless, LSTMs are still a form of RNN, and therefore suffer from the lack of parallelization and limited memory.

The Transformer architecture was introduced in 2017 for machine translation (Vaswani et al. 2017) but has achieved high performance across several domains. Notably, time series forecasting (Wu et al. 2020); general natural language processing tasks with BERT (Devlin et al. 2018); and computer vision with ViT (Dosovitskiy et al. 2020). Transformer models are an attention based model and do not use recurrence or convolutions in their design. The vanilla transformer uses both an encoder and decoder to model seq2seq problems. The encoder takes a sequence as input and outputs a latent representation of the sequence. The decoder takes this latent representation and outputs the desired sequence. However, it is possible to modify the design or use just the encoder or decoder to model seq2vec or vec2seq problems.

The multi-head self-attention mechanism which forms the basis of the model is found in both the encoder and encoder. The attention mechanism learns specific relations between each input and the other inputs in a sequence. It differs from convolutional layers as it considers all inputs and not just a window. It also differs from recurrent networks, as it processes the whole sequence at once rather than recurrently. Overall, the transformer model uses several self-attention mechanisms in parallel, known as multi-head self-attention, so that different dependencies can be learned independently, e.g. long term vs short term dependencies. The outputs from each head are concatenated and sent through a feed-forward network to gain the desired output shape. Finally, to prevent the model from being invariant to input permutations, positional encoding are added along with the input sequence. Vaswani et al. (2017) used periodic sinusoidal positional encoding but this is often changed depending on the domain.

4 AR property evolution

4.1 Dataset

In this section, we explore the utility of LSTM's and transformer architectures for predicting AR properties. The main feature of interest is the total unsigned magnetic flux:

$$\Phi = \sum |B_z| \, dA \,, \tag{1}$$

where B_z is the component of the flux in the direction of the line of sight and the sum is over the pixels covering the AR. Since energetic events such as flares and coronal mass ejections are generally accompanied by strong magnetic fields, Φ is a good indicator for such events.

In order to predict the evolution of Φ for a particular AR, we create a dataset consisting of sequences. Each time step in the sequence is attributed with several AR properties, including Φ itself, obtained from datasets collected from the Space-weather HMI Active Region Patches (SHARP; Bobra et al. 2014). We include the other properties because they might be significantly correlated to Φ and could help predict its evolution. The SHARP data we have used consists of observations for over 7000 ARs observed on the Sun between 2010 and 2022. These are divided randomly among training (70%) and validation (30%) data. Each AR is associated with a sequence of observations, which are separated by a 12-minute interval. The total number of observations for every AR is thus proportional to the duration of the AR. For each AR with sufficient observations, we construct multiple sequences of 300 time steps (60 hours), shifted by 20 time steps (4 hours). In order to simplify the task of the model, we reduce the number of time steps in each sequence from 300 to 60. For every 5 subsequent 12-minute observation, we sample an estimate for the AR properties in a 1-hour interval, resulting in 60 1-hour time steps. Next, we take the logarithm of features which span several orders of magnitude and finally, we standardize all of the features.

We divide each sequence into two sub-sequences: a *source*, which serves as the input to the prediction model, and a *target*, which the model is trained to predict. The first 40 time steps are used as the source and the last 20 are used as the target. For the source, we use 10 properties related to the physical features of the AR, whereas we only predict the flux Φ of the target.

4.2 Model Architectures

To predict the AR magnetic flux, we have experimented with LSTM and Transformer architectures.

4.2.1 LSTM

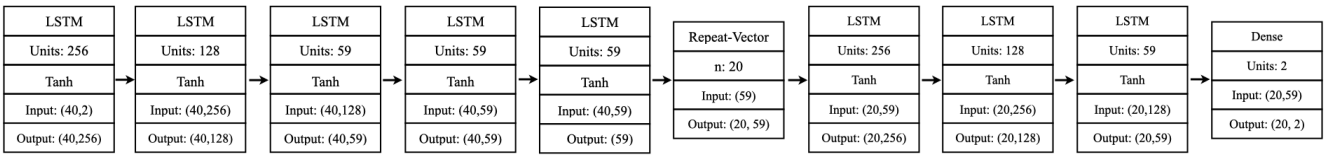

Figure 3: LSTM Architecture used to predict the magnetic flux of the Active Region (AR) with time.

For LSTMs, we initially trained the network using 10 physical AR features. However, after experimentation, we obtained the most consistent results by just using 2 features: the AR Magnetic Flux (the feature of interest) and the area of the AR. The architecture of the LSTM that we have developed is illustrated in Figure 3. It consists of 5 LSTM layers with the Tanh activation function, followed by a Repeat Vector layer that repeats the input 20 times corresponding to forecasting the 20 future time-steps, followed by 3 more LSTM layers, and an output dense layer with a total of $\sim 10^6$ trainable parameters. We use Adam with Nesterov momentum (Nadam) as the optimizer with a learning rate of 0.001 and the Mean Squared Error (MSE) as a loss function. The network

was trained for 50 epochs as further training caused it to overfit. We have attempted to decrease the overfitting by implementing dropout layers, using different resampling methods, normalizing the data instead of standardization, and augmenting the data by adding Gaussian noise. However, all attempts led to inferior results.

4.2.2 Transformer

For developing a Transformer-like architecture we primarily took inspiration from two sources. The first is a GitHub repository[1] with experiments for using a Transformer-based architecture for a TSF problem. The second is a paper by Wu et al. (2020) on predicting the prevalence of influenza-like illnesses with a Transformer-based model. From the first source, we adopt the idea of using only the encoder from the original Transformer and replacing the original decoder with one linear layer, which receives the flattened output of the encoder as input. From the second source, we use the idea of a time delay (TD) embedding, where the features of $N_{\text{TD}} - 1$ previous time steps are concatenated to each time step. For example, if $N_{\text{TD}} = 4$, the features of time step 10 become the concatenated features of time steps 7, 8, 9 and 10, the features of time step 9 become the concatenated features of time steps 6, 7, 8, and 9, etc. Using TD embedding outperformed the positional encoding of the original Transformer. In our case, we concatenate the 7 previous steps ($N_{\text{TD}} = 8$) to each time step, resulting in 8×10 features in total. As the first 7 time steps have less than 7 previous time steps, we discard them from the sequence, so the source consists of $40 - 7 = 33$ steps after our TD embedding. These concatenated features are then mapped to a hidden dimension with a single linear layer and subsequently fed to the encoder, which consists of 3 Transformer encoder layers. The features that constitute the input to the encoder thus carry information of the recent history in the sequence. For the encoder, we use a model dimension of 128 and 8 heads for the multi-head attention layers. Finally, we add a baseline to the output of the model, which consists of the magnetic flux feature value of the last time step of the source sequence. The model therefore has to predict the difference with respect to the baseline, instead of the magnetic flux feature value. The full architecture is given in Fig. 4.

4.3 Results

The performance of the LSTM architecture and Transformer-based architecture for predicting the magnetic flux are given here. We use the mean squared error (MSE) to evaluate the prediction accuracy. For a baseline prediction, we take the feature value of the magnetic flux of the last time step of the source and use this constant value as the prediction for each time step in the target. This results in a baseline MSE of 0.037.

4.3.1 LSTM

The LSTM achieved an MSE of 0.02 on the training set and 0.023 on the validation set, which is significantly lower than the baseline MSE of 0.037. The results are illustrated in Figure 5, where we can see that the predictions correctly follow the trend for the target, implying that the LSTM

[1] https://github.com/oliverguhr/transformer-time-series-prediction

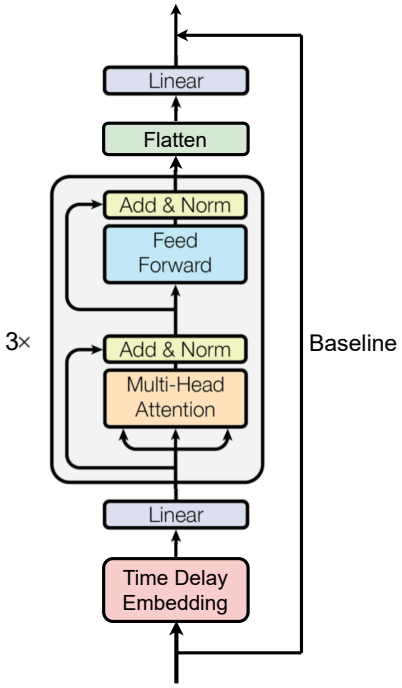

Figure 4: Architecture of the Transformer-based model used for predicting the AR total unsigned magnetic flux.

was able to learn the general "direction" of the data. However, the predictions are smooth and thus fail to learn the fluctuations in the data. As a result, the network's predictions can completely deviate from the target when it contains extreme fluctuations, as shown in set 10531.

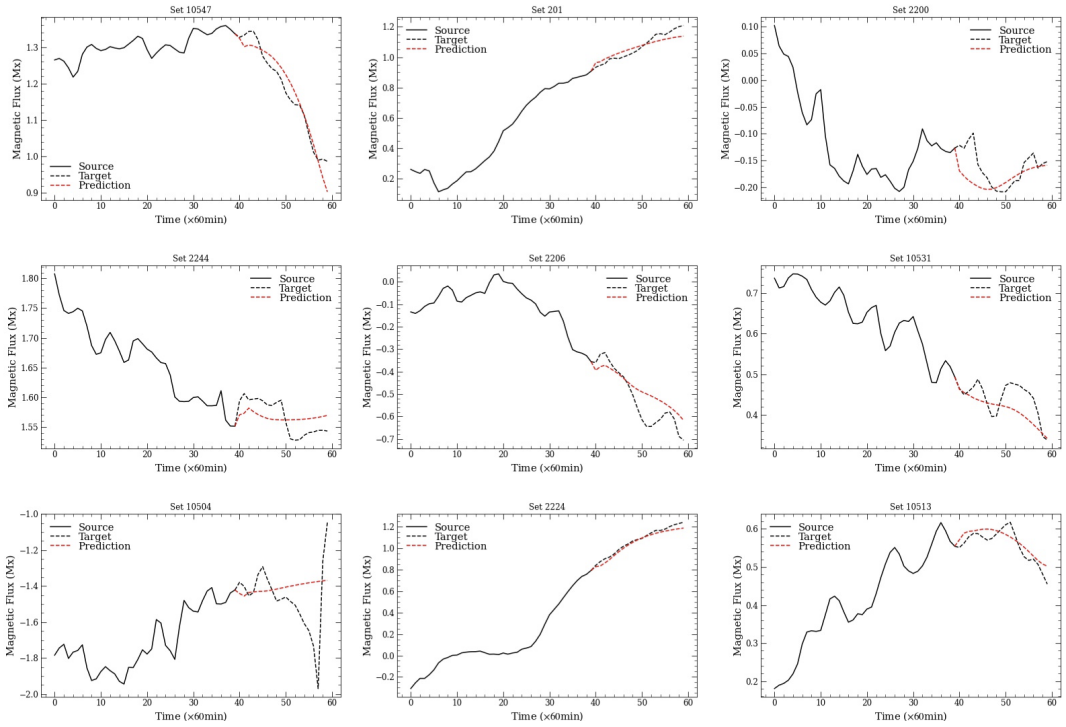

Figure 5: Predictions of the AR magnetic flux targets for 9 sequences as determined by the LSTM model.

4.3.2 Transformer

The performance of the Transformer-based model is similar to that of the LSTM model. We obtain a MSE of 0.019 for the validation set, which is slightly lower than that of the LSTM, but still significantly lower than the baseline MSE. The same targets predicted for the LSTM model in Figure 5 are predicted by the Transformer-based model in Figure 6. Comparing the predictions from both models, we see that the predictions from the LSTM model are generally smoother and seem to be better in following the trend. In contrast, the predictions from the Transformer-based model sometimes do not follow the trend as closely, but, as shown by e.g. set 10531, are generally better at predicting smaller fluctuations.

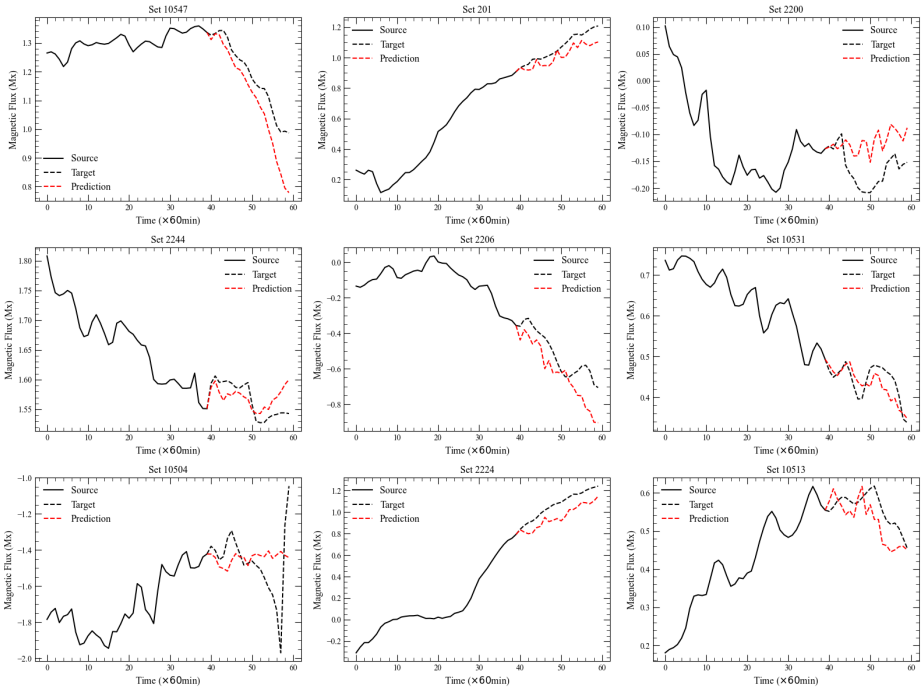

Figure 6: Predictions of the AR magnetic flux targets for 9 sequences from our Transformer-based model.

5 Flare predictions

5.1 Dataset

Following the work of Liu et al. (2019), our intention is to use sequential observations of an AR to predict its future flaring activity for which the use of various deep learning methods are ideally suited. Specifically, we consider the following classification problem: Within the next 24 hours, will this AR produce a solar flare and how energetic is the event? To quantify each event, classes are defined based on the magnitude of the X-ray flux produced by a given flare. Therefore, we distinguish between B ($10^{-7} - 10^{-6}\,\mathrm{Wm}^{-2}$), C ($10^{-6} - 10^{-5}\,\mathrm{Wm}^{-2}$), M ($10^{-5} - 10^{-4}\,\mathrm{Wm}^{-2}$) and X ($> 10^{-4}\,\mathrm{Wm}^{-2}$) classes, and any flux below $10^{-7}\,\mathrm{Wm}^{-2}$ is treated as regular solar activity.

To perform solar flare predictions, we consider two groups of predictive parameters. The first consists of the 25 physical parameters which are commonly used to characterize AR magnetic

Table 1: Distribution of the 10 hour time series used for flare forecasting. A large class imbalance is noticeable.

No Flare	B Flare	C Flare	M Flare	X Flare
94730	16571	26973	4555	450

field properties (Bobra & Couvidat 2015). This data is retrieved from the Space-weather HMI Active Region Patches (SHARP; Bobra et al. 2014). The second group corresponds to 15 features related to the flaring history. Among those, 6 are related to time decay values and are computed according to the formula of Wang et al. (2020). These parameters serve as a measure of the time elapsed between consecutive flare events. The remaining 9 parameters describe the total amount of flares having been observed in an AR over the last day or within its entire lifetime. This data is obtained from the *GOES* X-ray flare catalog provided by the National Centers for Environmental Information (NCEI) by selecting flares with identified AR's listed in the SHARP dataset. A summary of all 40 properties is given in Table A1. In this work, we decided to use 10 hours observations of an AR to perform the flare prediction. As such, 10 samples of the 40 AR properties at a cadence of 1 hour are used to complete the task. The duration of observation as well as the temporal separation of consecutive data samples is based on the work of Liu et al. (2019) and is somewhat arbitrary and may affect the final results. At the end, our dataset consists of 48549 time series which resulted in a flare in the next 24 hours while 94730 which remained quiet. The exact distribution of the data is summarized in Table 1. When creating the training, validation and test set, to minimize the degree of correlation between them, the three sets were sampled such that they cover different years. Data samples during the years 2010-2013 are used for training, those from 2014 are used for validation and finally years 2015-2018 form the test set. One concerning property is the huge class imbalance in our dataset, how we attempt to circumvent this problem will be discussed in the following section.

5.2 Model Architectures

The deep learning models investigated in this work consist of long short-term memory (LSTM) networks as well as transformers. For any architecture, the input sequence is of dimension 10×40 translating to the 40 AR properties at a cadence of 1 hour. The output consists of a one-hot encoding vector which describes the future flaring activity of the AR. We developed three LSTM and three transformer models of the same form but for varying classification tasks respectively. For each, the classes are defined differently. We consider a 5 classes (no flare, B, C, M, X), 3 classes (no flare, B+C, M+X) and 2 classes (flare, no flare) problem. Before feeding the data to the model, each individual property is normalized to ensure smoother training. Data standardization was also attempted, though this did not improve the final results.

5.2.1 LSTM

Figure 7 shows a sketch of a LSTM architecture used in this work. The model in itself is simple, consisting of a total of $\sim 4 \cdot 10^5$ trainable parameters. Each LSTM layer consists of 256 units and the standard hyperbolic tangent activation function is used Greff et al. (2017). Dropout layers are employed to reduce the effect of overfitting and the softmax activation function is applied to the

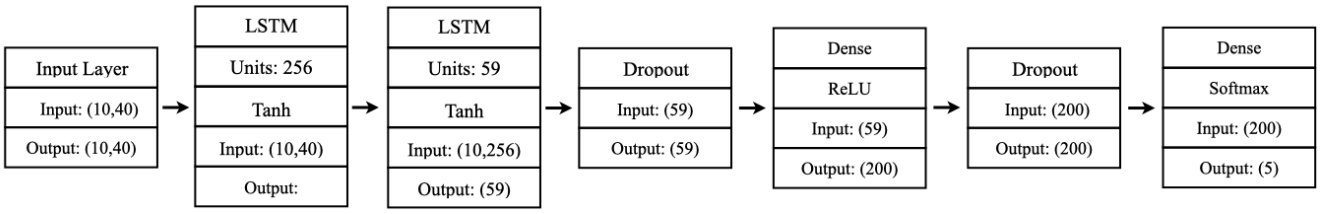

Figure 7: LSTM architecture for predicting AR flaring activity. Training is performed using stochastic gradient descent with an exponential learning schedule and a custom macro-F1 loss function to overcome the class imbalance.

dense output layer as needed for the one-hot encoding. For the LSTM model, stochastic gradient descent with an exponential learning schedule is used as an optimizer. This choice may come as a surprise, but the application of more prominent optimizers like 'adam' or 'nadam' resulted in training converging only after ~ 3 epochs still at a high loss. This observation suggests that the model got stuck in a local minima which alternatively could have been solved using a different weight initialization technique.

As alluded to before, one of the main problems of the dataset is the large class imbalance. This suggests that standard classification loss functions such as categorical cross entropy should not be employed for this problem given their negligence for class imbalance. One metric which is an accurate measure for an imbalanced dataset is the F1 loss. Hence, we implemented a custom loss function which aims to maximize the macro-F1 score. However, the macro-F1 is non-differentiable which means that in its true form it cannot be used as a loss function. One way to make it differentiable is by treating the values of true positive (TP) and false negative (FN) as probabilities instead of 0 and 1 integers. As an example, for a ground truth of 1 and model prediction 0.4, the customized loss function uses TP $= 0.4$ and FN $= 0.6$. Mathematically, we can write the customized loss function as:

$$\text{Loss} = 1 - \frac{1}{N}\sum_{i=1}^{N} F_{1,i}\,, \tag{2}$$

where $F_{1,i}$ is the F1-score of class i. Implementing this loss function proved the most effective to overcome the problem of class imbalance using LSTM architectures. Other methods were attempted to circumvent this issue such as class weighting which causes the model to "pay more attention" to examples from an under-represented class during training. Another approach consisted in data augmentation where Gaussian random noise was added to the data in an attempt to create a uniform distribution of class samples. However both of these methods did not yield any improvements. In future work a more sophisticated data augmentation technique could be implemented where instead of Gaussian noise the true instrumental error is added to the data.

5.2.2 Transformer

As we are not attempting to predict a sequence of data, our architecture does not feature decoder layers, unlike the original transformer model by Vaswani et al. (2017). After the model has formed a latent representation of the input, it forwards this directly to the classifier part. We use a sinusoidal positional encoding instead of `Time2Vec` (Kazemi et al. 2019), as this gives slightly better results.

Table 2: Model features of the transformer based models used to perform flare prediction.

Model	**AL-BERTO**	**BERT-mini**
Parameters	$\sim 2 \cdot 10^6$	$\sim 1 \cdot 10^7$
Heads	4	4
Hidden dimension	64	256
Encoding layers	4	4

For our experiments we followed two slightly different approaches: using a pre-made BERT-like model for sequence classification and developing a custom transformer-based model, which we name AL-BERTO. BERT is quite expensive to train from scratch, so for our experiments we opted to use an untrained BERT-mini[2] model, presented in Turc et al. (2019) and Bhargava et al. (2021). To code, train and evaluate our models we used the `PyTorch`, `transformers` and `scikit-learn` (Pedregosa et al. 2011) libraries. Besides having slightly different model parameters[3] (see Table 2), the main difference between the two models is the input masking operated in BERT. The reasoning behind randomly masking certain tokens is similar to the one that led to the introduction of dropout: it forces the model to take into consideration different relations between the elements of a sequence, eventually leading to a better learning (for an analysis of the masking percentages see Wettig et al. 2022). After the encoding layers, AL-BERTO features a simple classifier consisting of one `Flatten` layer (dim $= 64 \times 10$) and two hidden `Linear` layers with the same dimension. The final layer is another `Linear` layer with dimension equal to the number of labels to classify.

Model training was performed with `Adam` (Kingma & Ba 2014) using a decaying learning rate ($lr = 2 \times 10^{-5}$, $\gamma = 0.97$) in addition to *weighted cross-entropy*, due to the class imbalance. For this architecture, this choice yielded better results compared to using a continuous F1-score (as was used in Section 5.2.1), although after training the F1-score is still used to evaluate the models.

5.3 Results

To compare our deep learning models to conventional machine learning methods, we implemented a simple XGBoost model to serve as a baseline for how well our LSTM and transformer models perform. A summary of all the macro F1-scores of the models is given in Table 3.

5.3.1 LSTM

The results for each classification task are given in Table A2, A3 and A4 (column **L** represents LSTM models). The macro-F1 scores obtained when treating the problem in its most simple form (2 classes) is F1 = 71% for the test set, matching the performance of the baseline XGBoost model. Figure 8, illustrates the receiver operating characteristic (ROC) curve of the binary classifier with an area under the curve score (AUC) corresponding to AUC = 0.79 which implies a moderately good classification ability. However, when increasing the complexity by treating flare classes indi-

[2]From `https://huggingface.co/`

[3]For the internal parameters of BERT-mini see here

Table 3: Macro F1-scores (%) of the flare prediction models.

# classes	**XGBoost**	**LSTM**	**BERT-mini**	**AL-BERTO**
2	71	71	**73**	72
3	54	60	**61**	56
5	35	**45**	41	39

vidually, the macro-F1 score decreases with F1 = 60% for 3 classes and F1 = 45% for 5 classes. However, it is at this point where improvements over the simple XGBoost model become apparent thus suggesting that LSTM models perform better in understanding flare types. When analyzing the confusion matrix for the 3- and 5-class models (Figures B.3 and B.4), it is noticeable that each model is best in classifying no-flaring active regions and problems arise in the differentiation of flares of varying energies. Importantly, it is recognizable that the flare misclassification dominantly occurs between classes of similar energy. For instance, X flares ($> 10^{-4}\mathrm{W/m^2}$) which are the most energetic, are only misclassified as M flares ($10^{-5}-10^{-4}\mathrm{W/m^2}$), the model never confuses them with less energetic solar activity. Hence, the misclassification makes physical sense which suggests that the multi-class LSTM models have the ability of predicting solar flares within an energy interval (units $\mathrm{Wm^{-2}}$) of approximately 4 orders of magnitude. The exact origin of the misclassifications is unclear and requires further investigation. We suggest that it is likely that samples of similar energy outputs exhibit comparable flaring histories thus triggering the confusion.

5.3.2 Transformer

The confusion matrices of the best performing runs for AL-BERTO and BERT-mini, based on F1-score, for 3- and 5-classes are shown in Figure B.5 to B.8. For the binary classification case, the ROC curves are plotted in Figure 8. Depending on the number of labels, we observe different performances between our transformer model (AL-BERTO) and BERT-mini, as reflected by Tables A2, A3 and A4. In the most simple form (2-classes), BERT-mini outperforms AL-BERTO, both in AUC score (0.80 and 0.79) and macro average F1-score (73% and 72%). This is explained by the larger size of BERT-mini, having a number of trainable parameters one order of magnitude larger than our model. The misclassifications happen mostly in the *flare* label which is a consequence of the dataset imbalance. However, the implementation of weighted cross entropy improved the accuracy within this label from $\sim 50\%$ to $\sim 70\%$.

For three classes, BERT-mini still outperforms AL-BERTO in F1-score (61% and 56%) and most misclassifications happen in the most energetic flare types (M+X). The explanation for this behaviour is the same for the 2-class case, BERT-mini is a larger model and the M+X class type is underrepresented. When considering 5-classes, the result is the opposite; AL-BERTO outperforms BERT-mini according to the F1 score (42% and 39%). Observing the F1 scores relative to each label, we can see that BERT-mini performs better in each label besides the most energetic one (X flares), where it failed to correctly classify all the data. This is the reason behind the worse performance, which can be explained by the limited data for this category. BERT-like models are in general very data hungry to correctly gain general context and positional awareness and in this

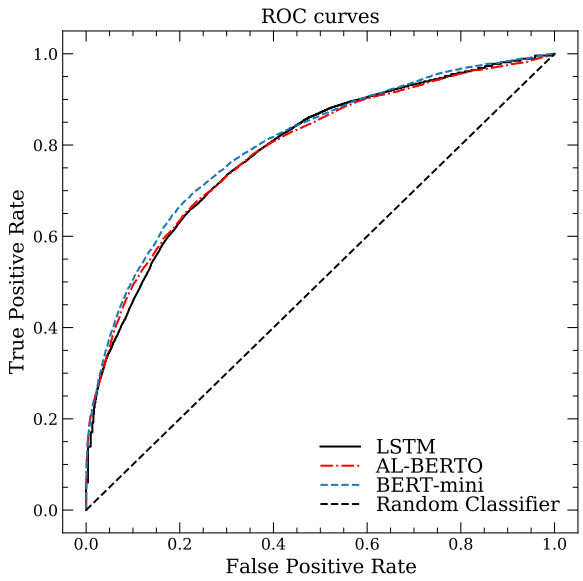

Figure 8: ROC curves for the binary classifier LSTM (solid black), AL-BERTO (dashed-dotted red) and BERT-mini (dashed blue) models. The classes are *flare* and *no flare* and the AUC scores are 0.79, 0.79 and 0.80 respectively.

case the X flare data is too sparse to achieve that. On the other hand, AL-BERTO was able to correctly classify some of the X flare data, although not with a high F1 score (only 30%).

For all cases, we can observe that the best performing label is always *no flare*, mainly due to the overabundance of such data samples. To better account for this in future works, data augmentation techniques incorporating instrumental noise should be applied. To improve classification, the addition of the label token in the input, as done by Dosovitskiy et al. (2020), could be considered. BERT-mini implements input masking, though it is unclear if this has a substantial effect on the performance. Input masking could be implemented in the future in AL-BERTO and more extensive tests might be carried out to determine its effect on the classification performance.

6 Conclusions

A number of areas related to deep learning were explored and applied to the field of solar physics within this project. Initial plans to further the work by Armstrong & Fletcher (2019) had to be abandoned due to the discovery that the dataset used within their study was heavily correlated, and attempts to salvage it using a variety of data augmentation techniques proved ineffective. We have reached out to the author to inform him of this issue. As a result, our plans were redefined with a focus on time-sequenced predictions of active region properties and solar flare activity. We addressed these two problem-cases using both LSTM and Transformer architectures, the primary results and conclusions of which are summarized hereinafter.

The LSTM designed to predict the evolution of complex AR properties over time outperformed that of our baseline model. The predictions correctly followed the trend for each target and it could

be concluded that an LSTM architecture performs well in capturing the general direction of the data. However, the model fails to capture small scale fluctuations and instead predicts evolutionary tracks that are smoothed. This makes it difficult for the LSTM to predict the evolution of properties that vary on short timescales. When our Transformer model was applied to this problem case, it performed slightly better than our LSTM model (and hence, better than the baseline model). Interestingly, the Transformer was better at capturing small scale fluctuations within the data compared to the smoother predictions captured by our LSTM. However, the Transformer was generally worse at predicting the overall trend direction compared to our LSTM. From a physical and space weather forecasting point-of-view, predicting the overall direction is more beneficial and hence it can be concluded that our LSTM model is better suited for this problem.

LSTM and Transformer models were built to predict if a specific AR would produce a solar flare over a 24 hour time-range. An untrained BERT-mini Transformer was also employed from Turc et al. (2019) and Bhargava et al. (2021) for further investigation. Firstly, architectures for each model were designed to predict flare occurrences in a binary fashion. These were then extended to include a classification of the predicted flares based on their strength, leading us to design three different architectures comprising of 2-, 3-, and 5-classes. For each model, all macro average F1-scores outperformed those of our baseline model. While BERT-mini was best at predicting and classifying flares into 2- and 3-classes, our LSTM model was best for the 5-class case. Hence, it can be concluded that LSTMs are better for this problem when extended to include more detailed classification. From a physical standpoint, the prediction of whether or not a flare will occur is more important than further classifying its magnitude. The performance of the binary prediction model for the three different architectures was therefore further analyzed using receiver operating characteristic (ROC) curves, and the BERT-mini model was again shown to tackle this problem best. Furthermore, it is clear from the ROC curves that the networks are not simply randomly assigning classes, where the AUC values convey reasonably good prediction capabilities.

From the work presented here, it is clear that both LSTMs and Transformers can be utilized to predict the evolution of AR properties over time. This has great practicality for space-weather forecasting efforts, as the monitoring of such properties is important for determining if eruptive solar activity will occur. Furthermore, it has been shown that deep learning methods are significantly better than traditional algorithms for detailed automated flare classification. Such applications can be of great use when addressing the data challenge existing in the field of solar physics. Further work could include applying data augmentation techniques (such as the inclusion of instrumental noise) to the AR property data to address the large imbalance within the dataset, especially to help improve performance for the 3- and 5-class models. To improve overall prediction, further analysis of which AR properties are most relevant for flare production should be explored. Finally, this work could be extended to include predictions of coronal mass ejections. These more extreme and potentially hazardous forms of solar eruptions are rather poorly understood, and deep learning provides the opportunity to both predict their occurrence and discover any correlations between their AR properties. The networks designed in this study can be found on our Github repository.

References

Armstrong, J., & Fletcher, L. 2019, Sol. Phys., 294, doi: 10.1007/s11207-019-1473-z

Bhargava, P., Drozd, A., & Rogers, A. 2021, Generalization in NLI: Ways (Not) To Go Beyond Simple Heuristics. https://arxiv.org/abs/2110.01518

Bobra, M. G., & Couvidat, S. 2015, ApJ, 798, 135, doi: 10.1088/0004-637X/798/2/135

Bobra, M. G., Sun, X., Hoeksema, J. T., et al. 2014, Sol. Phys., 289, 3549, doi: 10.1007/s11207-014-0529-3

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv, doi: 10.48550/ARXIV.1810.04805

Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. 2020, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, arXiv, doi: 10.48550/ARXIV.2010.11929

Fang, Y., Cui, Y., & Ao, X. 2019, Advances in Astronomy, 1, doi: 10.1155/2019/9196234

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. 2017, IEEE Transactions on Neural Networks and Learning Systems, 28, 2222, doi: 10.1109/tnnls.2016.2582924

Hanslmeier, A. 2007, Astrophysics and Space Science Library, Vol. 347, The Sun and Space Weather, 2nd edn. (Springer, Dordrecht)

Hochreiter, S., & Schmidhuber, J. 1997, Neural computation, 9, 1735, doi: 10.1162/neco.1997.9.8.1735

Kazemi, S. M., Goel, R., Eghbali, S., et al. 2019, Time2Vec: Learning a Vector Representation of Time, arXiv, doi: 10.48550/ARXIV.1907.05321

Kingma, D. P., & Ba, J. 2014, Adam: A Method for Stochastic Optimization, arXiv, doi: 10.48550/ARXIV.1412.6980

Liu, H., Liu, C., Wang, J. T. L., & Wang, H. 2019, The Astrophysical Journal, 877, 121, doi: 10.3847/1538-4357/ab1b3c

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, Journal of Machine Learning Research, 12, 2825

Turc, I., Chang, M.-W., Lee, K., & Toutanova, K. 2019, Well-Read Students Learn Better: On the Importance of Pre-training Compact Models, arXiv, doi: 10.48550/ARXIV.1908.08962

Vaswani, A., Shazeer, N., Parmar, N., et al. 2017, arXiv e-prints, arXiv:1706.03762. https://arxiv.org/abs/1706.03762

Wang, X., Chen, Y., Toth, G., et al. 2020, The Astrophysical Journal, 895, 3, doi: 10.3847/1538-4357/ab89ac

Wettig, A., Gao, T., Zhong, Z., & Chen, D. 2022, Should You Mask 15% in Masked Language Modeling?, arXiv, doi: 10.48550/ARXIV.2202.08005

Wu, N., Green, B., Ben, X., & O'Banion, S. 2020, arXiv e-prints, arXiv:2001.08317. https://arxiv.org/abs/2001.08317

A Armstrong

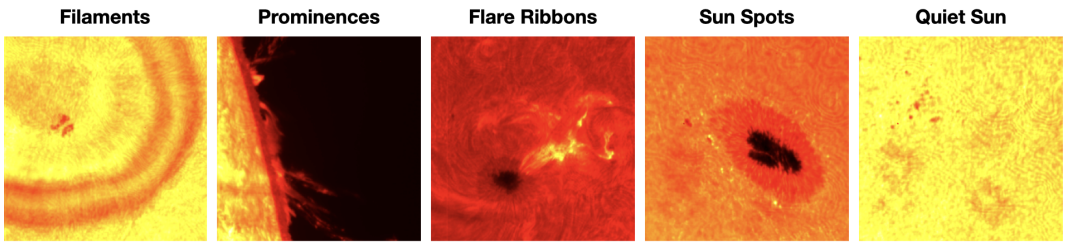

Figure A1: The five solar activity types for solar images: filaments, prominences, flare ribbons, sun spots, and quite sun.

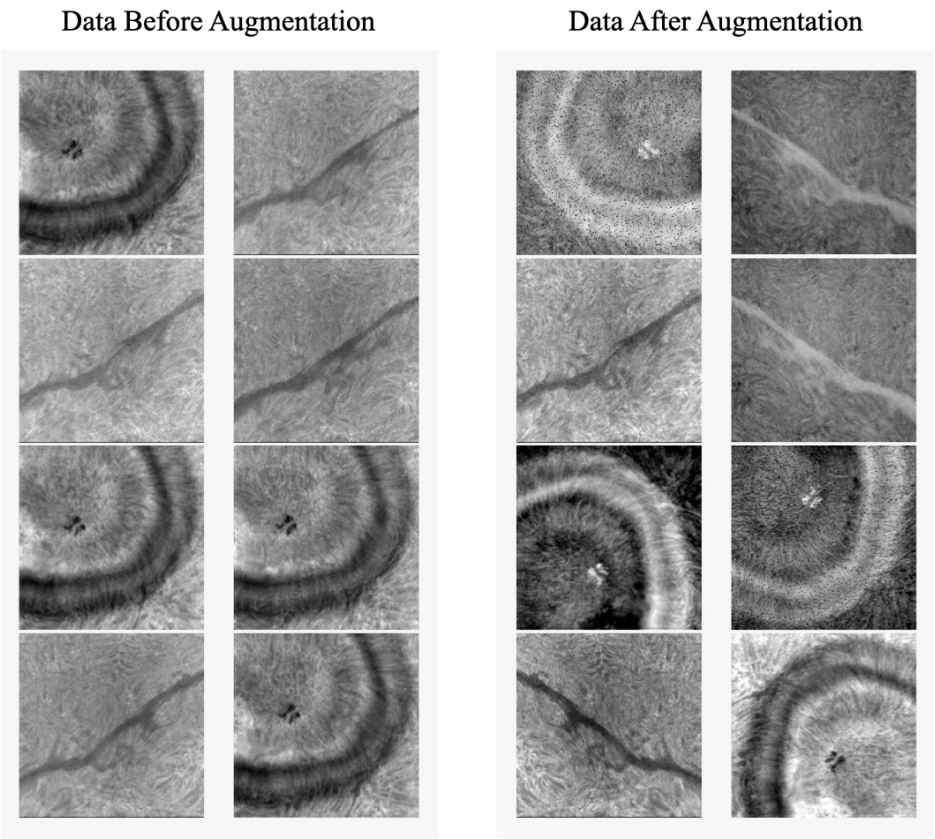

Figure A2: **Left panel:** randomly selected set of data, as we can see the images within each class are highly similar to each other.**Right panel:** the selected set after data augmentation which includes multiple functions such as; flipping, rotating, altering contrast, adding noise, and altering sharpness.

B Flare prediction

:

Table A1: Overview of the 40 features used for flare prediction including 25 magnetic parameters and 15 flare history features.

Keyword	Description	Formula
TOTUSJH	Total unsigned current helicity	$H_{c_{total}} \propto \sum \lvert B_z \cdot J_z \rvert$
TOTBSQ	Total magnitude of Lorentz force	$F \propto \sum B^2$
TOTPOT	Total photospheric magnetic free energy density	$\rho_{tot} \propto \sum (\boldsymbol{B}^{\mathrm{Obs}} - \boldsymbol{B}^{\mathrm{Pot}})^2 dA$
TOTUSJZ	Total unsigned vertical current	$J_{z_{total}} = \sum \lvert J_z \rvert dA$
ABSNJZH	Absolute value of the net current helicity	$H_{c_{abs}} \propto \lvert \sum B_z \cdot J_z \rvert$
SAVNCPP	Sum of the modulus of the net current per polarity	$J_{z_{sum}} \propto \lvert \sum^{B_z^+} J_z dA \rvert + \lvert \sum^{B_z^-} J_z dA \rvert$
USFLUX	Total unsigned flux	$\Phi = \sum \lvert B_z \rvert dA$
AREA_ACR	Area of strong field pixels in the active region	Area $= \sum$ Pixels
MEANPOT	Mean photospheric magnetic free energy	$\overline{\rho} \propto \frac{1}{N} \sum (\boldsymbol{B}^{\mathrm{Obs}} - \boldsymbol{B}^{\mathrm{Pot}})^2$
R_VALUE	Sum of flux near polarity inversion line	$\Phi = \sum \lvert B_{LoS} \rvert dA$ within R mask
SHRGT45	Fraction of area with shear $> 45^\circ$	Area with shear $> 45^\circ$/ total area
MEANSHR	Mean shear angle	$\overline{\Gamma} = \frac{1}{N} \sum \arccos(\frac{\boldsymbol{B}^{\mathrm{Obs}} \cdot \boldsymbol{B}^{\mathrm{Pot}}}{\lvert \boldsymbol{B}^{\mathrm{Obs}} \rvert \lvert \boldsymbol{B}^{\mathrm{Pot}} \rvert})$
MEANGAM	Mean angle of field from radial	$\overline{\gamma} = \frac{1}{N} \sum \arctan(\frac{B_h}{B_z})$
MEANGBT	Mean gradient of total field	$\overline{\lvert \nabla B_{tot} \rvert} = \frac{1}{N} \sum \sqrt{(\frac{\partial B}{\partial x})^2 + (\frac{\partial B}{\partial y})^2}$
MEANGBZ	Mean gradient of vertical field	$\overline{\lvert \nabla B_z \rvert} = \frac{1}{N} \sum \sqrt{(\frac{\partial B_z}{\partial x})^2 + (\frac{\partial B_z}{\partial y})^2}$
MEANGBH	Mean gradient of horizontal field	$\overline{\lvert \nabla B_h \rvert} = \frac{1}{N} \sum \sqrt{(\frac{\partial B_h}{\partial x})^2 + (\frac{\partial B_h}{\partial y})^2}$
MEANJZH	Mean current helicity	$\overline{H_c} \propto \frac{1}{N} \sum B_z J_z$
MEANJZD	Mean vertical current density	$\overline{J_z} \propto \frac{1}{N} \sum (\frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y})$
MEANALP	Mean characteristic twist parameter, α	$\alpha_{total} \propto \frac{\sum J_z B_z}{\sum B_z^2}$
TOTFX	Sum of x-component of Lorentz force	$F_x \propto -\sum B_x B_z dA$
TOTFY	Sum of y-component of Lorentz force	$F_y \propto \sum B_y B_z dA$
TOTFZ	Sum of z-component of Lorentz force	$F_z \propto \sum (B_x^2 + B_y^2 - B_z^2) dA$
EPSX	Sum of x-component of normalized Lorentz force	$\delta F_x \propto \frac{\sum B_x B_z}{\sum B^2}$
EPSY	Sum of y-component of normalized Lorentz force	$\delta F_y \propto \frac{-\sum B_y B_z}{\sum B^2}$
EPSZ	Sum of z-component of normalized Lorentz force	$\delta F_z \propto \frac{\sum (B_x^2 + B_y^2 - B_z^2)}{\sum B^2}$
Bdec	Time decay value based on the previous B-class flares only	$\mathrm{Bdec}(x_t) = \sum_{f_i \in F_B} e^{-\frac{t - t(f_i)}{\tau}}$
Cdec	Time decay value based on the previous C-class flares only	$\mathrm{Cdec}(x_t) = \sum_{f_i \in F_C} e^{-\frac{t - t(f_i)}{\tau}}$
Mdec	Time decay value based on the previous M-class flares only	$\mathrm{Mdec}(x_t) = \sum_{f_i \in F_M} e^{-\frac{t - t(f_i)}{\tau}}$
Xdec	Time decay value based on the previous X-class flares only	$\mathrm{Xdec}(x_t) = \sum_{f_i \in F_X} e^{-\frac{t - t(f_i)}{\tau}}$
Edec	Time decay value based on the magnitudes of all previous flares	$\mathrm{Edec}(x_t) = \sum_{f_i \in F} E_i \cdot e^{-\frac{t - t(f_i)}{\tau}}$
logEdec	Time decay value based on the log-magnitudes of all previous flares	$\mathrm{logEdec}(x_t) = \sum_{f_i \in F} \log(E_i) \cdot e^{-\frac{t - t(f_i)}{\tau}}$
Bhis	Total history of B-class flares in an AR	-
Chis	Total history of C-class flares in an AR	-
Mhis	Total history of M-class flares in an AR	-
Xhis	Total history of X-class flares in an AR	-
Bhis1d	1-day history of B-class flares in an AR	-
Chis1d	1-day history of C-class flares in an AR	-
Mhis1d	1-day history of M-class flares in an AR	-
Xhis1d	1-day history of X-class flares in an AR	-
Xmax1d	Maximum X-ray intensity one day before	-

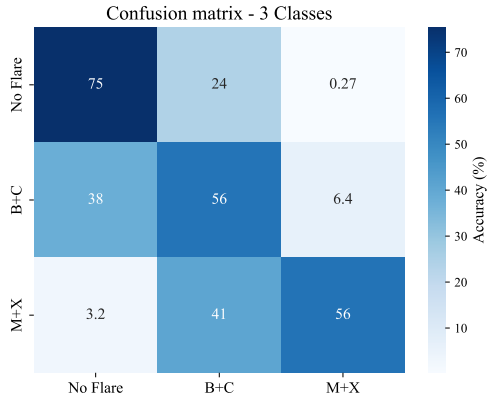

Figure B.3: LSTM model confusion matrix using 3 classes.

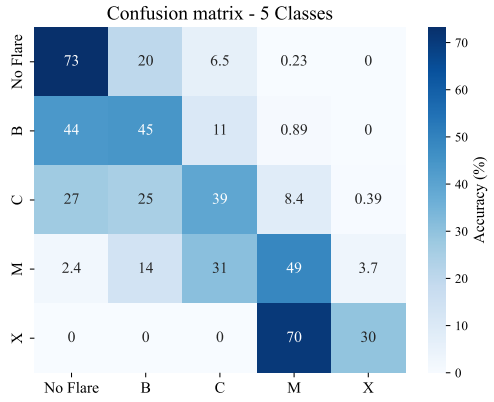

Figure B.4: LSTM model confusion matrix using 5 classes.

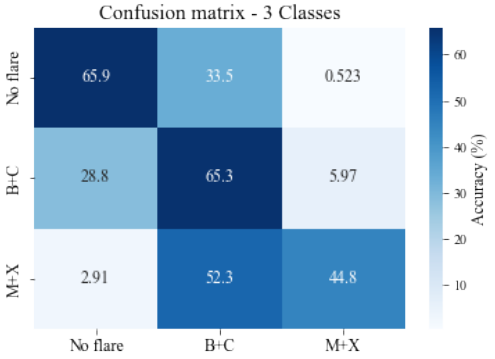

Figure B.5: AL-BERTO model confusion matrix using 3 classes.

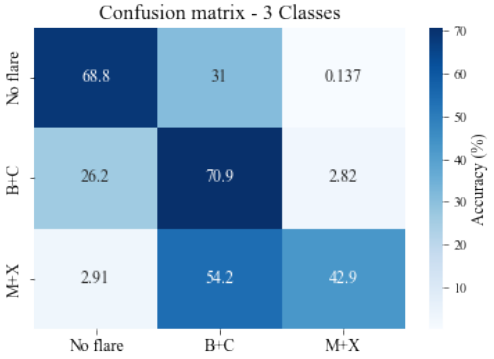

Figure B.6: BERT-mini model confusion matrix using 3 classes.

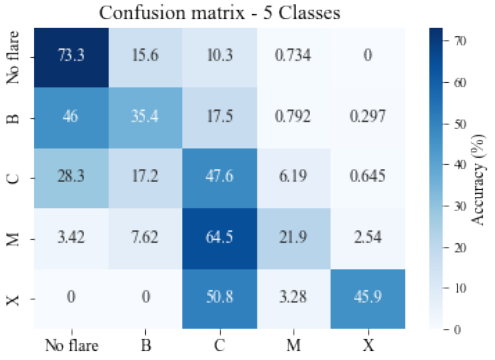

Figure B.7: AL-BERTO model confusion matrix using 5 classes.

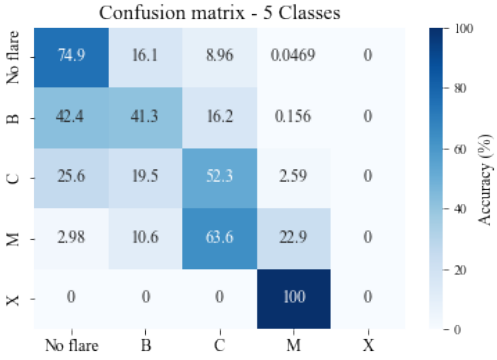

Figure B.8: BERT-mini model confusion matrix using 5 classes.

Table A2: Solar flare prediction results for models with *two* classes. **L** represents our LSTM model, **B** represents BERT-Mini, and **A** represents the results from our Transformer model (AL-BERTO model).

	Precision (%)			*Recall (%)*			*F1-score (%)*		
	L	**B**	**A**	**L**	**B**	**A**	**L**	**B**	**A**
No Flare	78	80	79	80	79	77	79	79	78
Flare	65	65	64	62	68	67	64	67	65
Macro Avg	71	**73**	72	71	**73**	72	71	**73**	72

Table A3: Solar flare prediction results for models with *three* classes. **L** represents our LSTM model, **B** represents BERT-Mini, and **A** represents the results from our Transformer model (AL-BERTO model).

	Precision (%)			*Recall (%)*			*F1-score (%)*		
	L	**B**	**A**	**L**	**B**	**A**	**L**	**B**	**A**
No Flare	78	83	81	75	69	66	77	75	73
B + C Flare	54	54	50	56	71	65	55	61	57
M + X Flare	41	54	36	56	43	45	47	48	40
Macro Avg	58	**64**	55	62	**61**	59	60	**61**	56

Table A4: Solar flare prediction results for models with *five* classes. **L** represents our LSTM model, **B** represents BERT-Mini, and **A** represents the results from our Transformer model (AL-BERTO model).

	Precision (%)			*Recall (%)*			*F1-score (%)*		
	L	**B**	**A**	**L**	**B**	**A**	**L**	**B**	**A**
No Flare	79	80	78	73	75	73	76	77	76
B Flare	31	34	32	45	41	35	36	37	34
C Flare	50	47	42	39	52	48	44	49	44
M Flare	43	50	27	49	23	22	45	31	24
X Flare	21	0	23	30	0	46	24	0	30
Macro Avg	**45**	42	40	**47**	38	45	**45**	39	42