Automatic Spread-F Detection Using Deep Learning

Christopher Luwanga[1], Tzu-Wei Fang[2], Amal Chandran[1], and Yu-Ju Lee[3]

[1]Nanyang Technological University
[2]CIRES-University of Colorado
[3]Cooperative Institute for Research in Environmental Sciences, University of Colorado Boulder

November 23, 2022

Abstract

Spread-F (SF) is a feature that can be visually observed on ionograms when the ionosonde signals are significantly impacted by plasma irregularities in the ionosphere. Depending on the scale of the plasma irregularities, radio waves of different frequencies are impacted differently when the signals pass through the ionosphere. An automated method for detecting SF in ionograms is presented in this study. Through detecting the existence of SF in ionograms we can help identify instances of plasma irregularities that are potentially affecting the high-frequency radio wave systems. The ionogram images from Jicamarca observatory in Peru, during the years 2008 to 2019, are used in this study. Three machine learning approaches have been carried out: supervised learning using Support Vector Machines, and two neural network-based learning methods: autoencoder and transfer learning. Of these three methods, the transfer learning approach, which uses convolutional neural network architectures demonstrates the best performance. The best existing architecture that is suitable for this problem appears to be the ResNet50. On a test set of 2050 ionograms the ResNet50 model provides an accuracy of 89 percent, recall of 87 percent, precision of 95 percent as well as Area Under the Curve (AUC) of 96 percent. In addition to the model, this work also provides a labelled dataset of over 28,000 ionograms, which is extremely useful for the community for future machine learning studies.

Hosted file

`essoar.10509820.1.docx` available at https://authorea.com/users/543826/articles/601417-automatic-spread-f-detection-using-deep-learning

Posted on 23 Nov 2022 — CC-BY 4.0 — https://doi.org/10.1002/essoar.10509820.1 — This a preprint and has not been peer reviewed. Data may be preliminary.

Christopher Luwanga[1], Tzu-Wei Fang[2], Amal Chandran[1,3], Yu-Ju Lee[4]

Automatic Spread-F Detection Using Deep Learning

[1] Nanyang Technological University Singapore.

[2] NOAA Space Weather Prediction Center, Colorado, USA

[3] University of Colorado Boulder

[4] Cooperative Institute for Research in Environmental Sciences, University of Colorado Boulder

Corresponding author: Christopher Luwanga (Luwa0001@e.ntu.edu.sg)

Key Points

1. Adopt machine learning techniques to detect Spread-F in ionograms
2. **Makes publicly available an annotated dataset with more than 28,000 ionograms**

Abstract

Spread-F (SF) is a feature that can be visually observed on ionograms when the ionosonde signals are significantly impacted by plasma irregularities in the ionosphere. Depending on the scale of the plasma irregularities, radio waves of different frequencies are impacted differently when the signals pass through the ionosphere. An automated method for detecting SF in ionograms is presented in this study. Through detecting the existence of SF in ionograms we can help identify instances of plasma irregularities that are potentially affecting the high-frequency radio wave systems. The ionogram images from Jicamarca observatory in Peru, during the years 2008 to 2019, are used in this study. Three machine learning approaches have been carried out: supervised learning using Support Vector Machines, and two neural network-based learning methods: autoencoder and transfer learning. Of these three methods, the transfer learning approach, which uses convolutional neural network architectures demonstrates the best performance. The best existing architecture that is suitable for this problem appears to be the ResNet50. On a test set of 2050 ionograms the ResNet50 model provides an accuracy of 89 percent, recall of 87 percent, precision of 95 percent as well as Area Under the Curve (AUC) of 96 percent. In addition to the model, this work also provides a labelled dataset of over 28,000 ionograms, which is extremely useful for the community for future machine learning studies.

INTRODUCTION

The ionosphere is a region in the upper atmosphere consisting of charged particles, e.g., ions and electrons. These particles are created when the neutral atmosphere is ionized by solar radiations and through collisions with other high

energy charged particles coming from the Sun. The number density of the charged particles exhibits strong diurnal variations as well as significant day-to-day variability (Fang et al., 2018). Typically, the ionosphere can be divided into several layers in altitude based on the profile of electron density (e.g., Bora, 2017). They are D (50-95 km), E (90-150 km), F1 (140-200 km), F2 (>200 km) layers. D, E, and F1 layers mostly disappear after sunset due to lack of photo ionization sources and strong recombination in the nighttime. In the D- and E-region altitude, the ionization is mostly caused by the solar X-ray. For the F-region ionosphere, extreme ultra-violet (EUV) dominates the process. The ionospheric density variations can significantly impact the radio wave communications that rely on reflection of radio waves or the higher frequency radio waves that transmit through the ionosphere. The gradients in the plasma density can also diffract radio signals and cause amplitude or phase fluctuation in the Global Navigation Satellite System (GNSS) signals.

In the ionosphere, plasma instability conditions often occur during the post-sunset period and lead to the formation of plasma bubbles or plasma irregularities. These bubbles and irregularities show strong density fluctuation and their movements are aligned with the background magnetic field structure (Li et al., 2021). When a wave traverses the ionosphere in the presence of such plasma irregularities, the wave experiences interference different to those when there is no plasma bubble (Sahai et al., 2004). The resulting impact on the signal amplitude and phase can be so significant that the information carried by the wave may be distorted or lost. Studying the ionosphere irregularities helps us to determine the spatial and temporal variations of these changes, estimate the impact of these fluctuations on radio signals, and mitigate their influence on technologies that we rely on..

Ionosonde is one important ground-based instrument that is used to determine ionospheric properties. The ionosonde consists of a suite of antennas coupled with analog and digital circuitry to generate and record signals of frequencies between 0.1 MHz and 30 MHz. The signal is sent primarily vertically into the ionosphere. Depending on the plasma density in the ionosphere, the signal with different frequencies get reflected back to the ionosonde (Kalita et al., 2019) at different points. When an ionosonde sends a radio wave of frequency ω the wave encounters a plasma of increasing density, and thus increasing plasma oscillation frequency (ω_p) because plasma frequency is proportional to plasma density, up to a point where $\omega = \omega_p$; at this point the wave from the ionosonde is reflected (Floer, 2020). On the other hand, when the incident wave frequency ω is higher than the plasma frequency ω_p the wave would penetrate through the ionosphere. Additionally, the presence of the Earth's magnetic field makes the ionosphere birefringent such that when one incident wave is reflected it is split into two waves, one of which is "Ordinary" or O, and the other is "eXtraordinary" or X. The O-mode is said to be "ordinary" since it behaves as though there were no magnetic field and reflects at the point in the ionospheric plasma where $\omega = \omega_p$. The X-mode waves do not reflect at $\omega = \omega_p$. Rather the X-mode reflects at

$\omega = \omega_p + \frac{\omega_g}{2}$, where ω_g is the electron-gyrofrequency (Floer, 2020). Because the two reflected X-mode waves now have different wavelengths, their propagation through the rest of the ionosphere will differ (similar to how *white* light splits in a prism due to dispersion) and will emerge at the receiving ionosonde as two separate waves (Maruyama, 2002)

The reflected wave amplitude is recorded by the ionosonde. The raw data can then be processed and is often presented in terms of a visual map called ionogram. Some examples of ionograms are shown in Figure 1. On a typical ionogram, the distance (virtual height) that the radio wave travels is shown on the y-axis whereas the frequency of the wave sent by the ionosonde is shown on the x-axis. The distance is calculated by multiplying the speed of light by the duration between the sending of the signal and its reception by the ionosonde. The distance is "virtual" because the light wave does not actually travel through the ionosphere at the speed that it does in vacuum; the speed of light in the plasma (which the ionosphere is) is less than in vacuum, therefore the actual distance is less than the virtual height.

The colors in the ionogram represent various information of the signal, such as the direction where the echo comes from, the Doppler shift, and the type of the echo (i.e., O-mode or X-mode). For example, in Figure 1, the red points are for a vertical echo of the ordinary mode and the deep blue is for the ordinary echo from the North-North West direction. The size (geometric area) of each rectangular point encodes the magnitude of the amplitude of the echo (Reinisch, 2009). The black curve is for the ion density profile derived from the automatic tracing algorithm called ARTIST version 4. ARTIST (Automatic Real Time Ionogram Scaler with True Height) is a software system for "scaling" ionospheric parameters—where "scaling" entails calculating or estimating the values for the parameters.. ARTIST provides the values for peak frequency in the F2 layer, foF2, the virtual height of the F layer, h'F, as well as other useful derivative information such as the Maximum Useable Frequency (MUF) to communicate with someone at a distance of 3000 km away, shown as MUF(3000). These parameters and values are shown in tabular form on the ionograms. The density profile above the ionospheric peak density is estimated by the ARTIST tool since the ionosonde can only detect echoes below the density peak.

Figure 1 shows a sample ionogram without SF (Normal) and one with SF, both from the Jicamarca radio observatory (JRO). The sample on the left is observed on 25 August 2013 (day number 237 in 2013) at 01:30 am Universal Time (UT). The one on the right was observed on 02 September 2013 (day number 245) at 01:30 am UT.

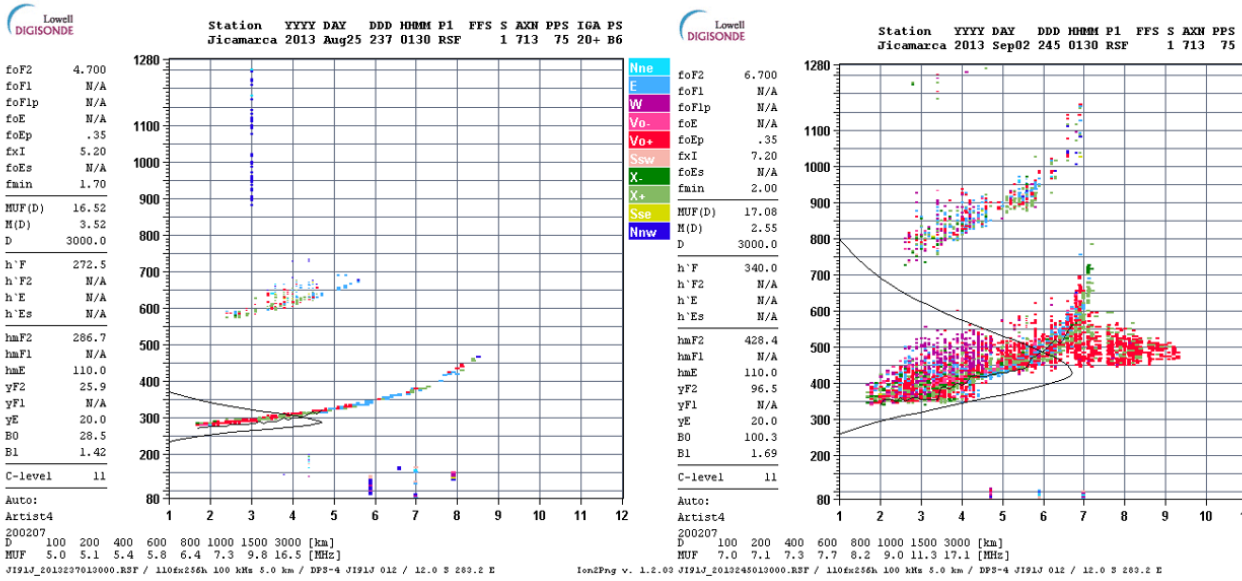

Figure 1: Ionogram without SF on the left and ionogram with SF on the right. The colors encode the following parameters: the direction from which the echo comes, the Doppler, and the mode of the echo—that is, whether it is an extraordinary mode or ordinary mode.

Multiple reflections commonly appear in an ionogram, which shows multiple echoes at multiple heights and can be easily identified. They are caused by multiple reflections of a signal bouncing between the ground and ionosphere (Maruyama, 2002). Another feature that is frequently shown in the nighttime ionogram is called Spread-F (SF). When a signal from an ionosonde is sent upwards, ideally it is expected that only one main echo is received. This echo provides the information about the height of the layer from which it was reflected. There are cases where *different* frequencies are reflected at the *same* height, rather than at different heights as is normally expected. These behaviors are associated with plasma irregularities in the ionosphere after the post-sunset period. When these cases arise, they show up on an ionogram as a spread in frequency or virtual height (range). The anomalies are thus called SF, owing to their spread appearance on ionograms—the echo is literally spread across many more pixels than those in normal cases. When many frequencies are reflected at the same height, this is called Frequency SF (FSF). When the same frequency is reflected at multiple heights, this is called Range Spread-F (RSF) (Bhaneja et al., 2009). Sometimes it is useful to divide SF into further categories: Strong Range Spread-F (SSF) and Mixed Spread-F (MSF) are two more ways to categorize SF. The type of Spread observed can aid in determining where in the ionosphere the irregularities are. For example, observing FSF is a sign that the density irregularities are close to the F peak region. RSFs are observed when the

irregularities (bubbles) are below the F peak height. When the irregularities are distributed randomly across the F region, MSFs are most likely to be observed. Finally, SSFs appear when the density depletion below the F peak manages to move beyond the F peak region and on to the top side (Alfonsi et al., 2013). In this study, the ionograms are divided into two groups: those without and those with SF, irrespective of the type of SF.

Another phenomenon associated with plasma bubbles and irregularities is ionosphere scintillation, which typically refers to signals from GNSS measurements. Scintillation represents the random fluctuation in the phase and amplitude of received GNSS signal (Cander, 2019). The signals propagate through the irregular plasma density structures and the diffraction and/or refraction leads to signal intensity drops and phase shifts. The returned signals then appear to be scintillated. Imagine a planar wave incident on two ionosphere patches of different densities, one patch of normal background density and another of much lower density. They emerge at different times because the speed of a wave in a given medium is a function of the refractive index of that medium; and the refractive index is a function of plasma density (Wiesemann, 2013). Therefore, the wave will move at different speeds in the low-density plasma bubble than in the normal density region. The difference in duration of travel (and thus phase at exit) will result in the vector sum of the two emerging waves being different than the amplitude of the original waves (e.g., Kintner et al., 2009; Maruyama, 2002). An instrument receiving the signals will see fluctuations in amplitude that are not initiated by the sender. The result is that a receiver may be unable to accurately extract the information that was sent. The GNSS satellites transmit the L-band signals (~1.575 GHz) at a height of about 20,200 km to receivers on the ground. The L-band signal has to cross the ionosphere, and is therefore affected by the conditions in the ionosphere. The scintillation in GNSS signal can lead to position errors with the severity of impact of these errors depending on the application (Kintner et al., 2009).

The identification of SF from ionogram is important because SF is a sign of a large-scale ionospheric irregularities (100s km). The evolution of plasma irregularities largely depends on the background ionosphere and thermosphere conditions and can potentially lead to a small-scale (10s km) irregularities that may affect communication and navigation systems. Several studies have shown that strong *correlation* between scintillation and spread F can be observed. Shi et al. (2011) showed that the occurrence rate of scintillation of the GPS L-band signal was high during times when SSF occurrence rate was also high at a low-latitude station of Hainan (109.1°E, 19.5°N; dip latitude 9°N). Liang et al. (2015) also showed similar conclusions using data from another low-latitude station at Vanimo (2.7°S, 141.3°E; dip latitude 11°S). Therefore, by identifying instances of Spread F it is possible to estimate the occurrence rate of scintillation events.

Identifying SF has been typically a manual process. Existing flags based provided by autoscaling software such as the ARTIST program do not provide

reliable results. Labelling all instances of SF remains challenging since it is a rather time-consuming process; therefore, establishing an automated method based on machine learning (ML) algorithm is the main goal of this study. A few generally data-driven and specifically machine learning methodologies have been carried out on SF identification. Scotto et al. (2018) developed a module for automatically detecting SF from the equatorial region. Their software module complements Autoscala, which is software developed for scaling the ionosonde data in order to get parameters of interest such as critical frequencies, heights, and maximum useable frequencies.. However, in the presence of SF, one can no longer trust the data collected. Scotto et al. (2013) therefore built a numerical model that identifies which data Autoscala must exclude. That is, once a SF has been detected the ionosonde data are no longer used for calculating ionospheric parameters. They used 198 samples from Tucuman station (26.9°S, 294.6°E) to determine a threshold for splitting between SF and no SF. After obtaining the appropriate threshold, they tested using 7,649 manually scaled ionograms for every hour of the year 2016. They obtained a true positive ratio of 81.55%.

Another model has been built with data from a low latitude station at São José dos Campos (23.2°S, 45.9°W, dip latitude 17.6°S) and an equatorial station at Palmas (10.2°S, 48.2°W, dip latitude 5.5°S). This model was based on fuzzy relation. The experimenters used 4320 ionograms collected over a period of 30 days between 6 pm and 6 am local time. The months studied are January and February 2004 which is summer for Brazil. The module was able to detect the presence of SF in about 92% of the cases that had been labelled as having SF (Pillat, Fagundes, & Guimarães, 2015). Another study has utilized the ML technique called decision trees to establish a model to identify Spread F (Lan et al., 2018). The inputs for the decision tree model are 47,711 ionograms recorded at Puer station (22.7°N, 101.5°E; Geomagnetic latitude: 12.8°N) in Yunnan province of China. The year 2015 was chosen in this study.

Data from the first half of every month—January to December—to build their decision tree, and then the data from the latter half of each was used for validation. In order to test their model, 41,471 ionograms from 2015 are used — the test ionograms were different from the training set. Then they also used 7,367 ionograms from July 2016 because this was a month with significant amount (1,428) of SF. This study obtained 89% accuracy on correctly identifying cases with SF.

These results show some success. However, the models have only been tested for the same location from which the data for training the model were obtained; the models thus may not work for other locations, such as Jicamarca, Peru. While Lan et al. (2018) attempted testing with data from a different year, only one month was tested. Machine learning models can be very dependent on the data used for training. This means that one can have an extremely good model based on carefully selected data. It is thus important to include data that is as temporally diverse as possible so that the model does not simply learn the features that apply for one particular year or season. In this work,

a large dataset with over 28,000 ionograms near equinoxes spanning multiple years (2008 to 2019) are utilized to provide a more comprehensive coverage for SF events.

It is worth emphasizing that the Jicamarca Radio Observatory (JRO, 12°S, 76.8°W, and dip latitude 1°N)) in Peru sits close to the magnetic equator and is the premier scientific facility in the world to study the equatorial ionosphere. JRO hosts multiple radars and ground-based instruments that are used to detect ionosphere properties. It has a collection of scatter radars consisting of three 1.5 megawatt transmitters and incoherent scatter radar (Milla, 2017). There are also many magnetometers distributed around Peru with which scientists can study changes in the magnetic field associated with ionospheric currents. Some of these measurements are extremely useful and can be combined with ionosonde data to facilitate extensive studies on ionosphere irregularities and the background conditions that lead to the irregularities. As mentioned before, none of the existing machine learning models were built or tested with ionograms from JRO. Therefore, establishing a machine learning algorithm that works with the long-term JRO data and making the labelled dataset publicly available are important objectives of this study. Following the steps described in this paper will also allow other researchers to easily build upon this work.

We describe the data used in this work in Section 2. Section 3 discusses the machine learning methods used in the study, namely Support Vector Machines (SVM) and the Convolutional Neural Networks (CNNs). The CNNs include an autoencoder architecture and other architectures based on ResNet50, VGG16, and InceptionV3 architecture from the public domain. The experiments and results are then shown in Section 4, together with a discussion of the results. In Section 5, key findings of the work and potential future work are summarized.

DATA

Various long-term observations from multiple instruments from JRO has provided tremendous information and novel insight for understanding the equatorial ionosphere. The ionosonde at JRO (station ID: JI91J) has a virtual height range of 80 km to 1320 km and a probe frequency range between 0.5 MHz and 20 MHz. There are two temporal data collection modes: one mode is to collect every 15 minutes, and the other is every 5 minutes. Data from both cadences are used in this study. However, note that in one of the experiments, the 5-minute cadence data were excluded in order to assess the impact that data temporal resolution has on the ML model.

The ionograms are obtained from a NOAA data repository (https://data.ngdc.noaa.gov/instruments/remote-sensing/active/profilers-sounders/ionosonde/). SF occurs more often during the chosen months of March, April, August, and September (Chapagain et al., 2009). All the ionograms used in this study are from the years 2008-2019 (for 2019 only March). The period between 2008 to

2019 corresponds to the solar cycle 24. The reason for choosing a season where SF is expected the most is to have sufficient data for both the Normal and SF classes. If the data were not so carefully chosen, there would be a bias since normally there are far more normal ionograms than there are those with SF.

SF is a primarily nighttime phenomenon (Chapagain et al., 2009), though it has also been shown to occur, in some rare cases, on the dayside at low-latitude regions (Jiang et al., 2016). Therefore, the ionograms from 19:00 local time (LT) of present day to 05:00 LT the following day are chosen in this study.

The ionograms were then manually classified and placed into three categories: Spread-F (SF), Normal (that is, no SF) and unsure. The unsure samples were those where it was not clear whether the sample contained SF or not. There are three distinct features of the unsure samples. The largest group is of those where there was too little information for even a human to make sense of it; there were either no echoes at all or just a very small number. The second group of unsure samples had a lot of other noisy echoes that made it hard for the human labeller to confidently determine whether it could be due to SF or not. Finally, a smaller group had echoes all over the place with no clear patterns. Since the unsure samples were hard to place into either Normal or SF class, they were not used for training, validation or testing. There were about 4236 unsure samples, compared to around 28 thousand used in training, validation and testing.

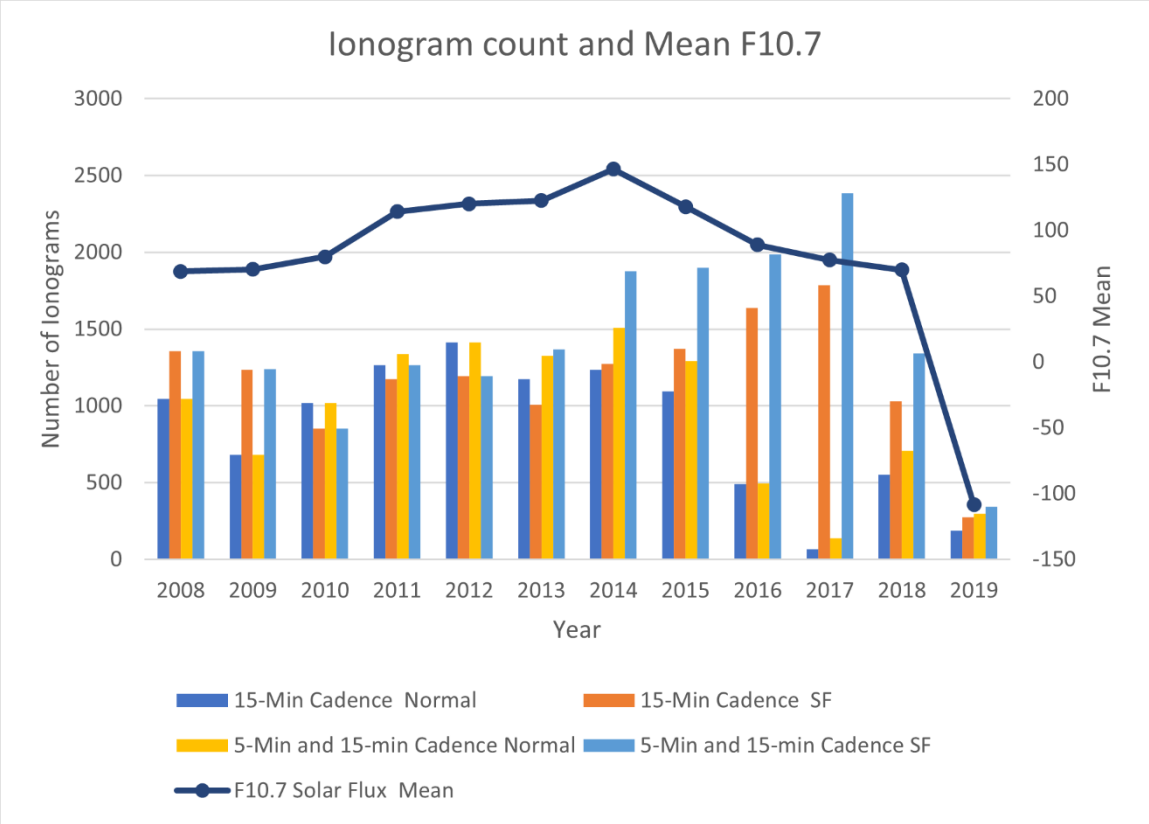

Figure 2: Number of ionograms from the SF class and Normal class as function of year. The set of ionograms of 15-minute cadence excludes ionograms collected every five minutes, while the 15-min and 5-min cadence set includes both the ionograms collected every 5 minutes as well as those collected every 15 minutes. The F10.7 index depicts the phases in the 24^{th} solar cycle.

Figure 2 demonstrates the numbers of samples in each year used in this study. Ionogram samples collected with 15-minute and 5-minute cadences are also shown in the figure. It shows that in some years, such as 2017 and 2018, there are more SF samples than normal. It should be noted that the months chosen are deliberately those with greater likelihood of SF; for example, the months of May and June have been shown to not have a lot of SF so they are omitted from the dataset. The dataset is also chosen to coincide with the 24^{th} solar cycle which was from 2008 to 2019, as illustrated by the F10.7 solar flux index. The F10.7 index results from calculating hourly average of the flux of radiation, whose wavelength is 10.7 cm, issuing from the sun (Petrova et al., 2021). F10.7 has a long historical record and has been a good proxy for Ultra-Violet radiation while also correlating with sunspot number. The F10.7 annual mean is shown in Figure 2 to show the different phases of the solar cycle 24 from which come

the ionograms used in this work.

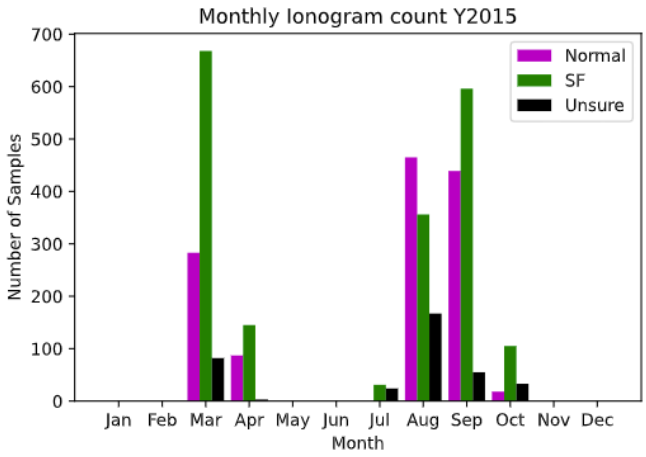

Figure 3: Monthly distribution of samples in year 2015 as divided into Normal (purple), SF (green) and Unsure (black). Unsure are not used in creating the models.

Monthly distribution of samples for the year 2015 is also shown in Figure 3. The month of March for 2015 has significantly larger number of SF samples (668) than Normal (283). In total, there are 1292 normal samples and 1901 SF samples. Such skewness in the data distribution, though mild in this case, is important to notice because it restricts what kinds of models one can use. As an example to show how data skewness can affect models, consider the case of a training dataset where 900 out 1000 ionograms are normal and the other 100 are SF. A simple classification ML model that always predicts that a given ionogram is in the normal class will have a 90% accuracy, which is largely due to the unbalanced training dataset. Thus, a well-balanced dataset is critical to establish a reliable ML model in this study.

The number of SF events as a function of time for day 94 of 2008, 2014 and 2018 are shown in Figure 4. For 2018 there was 1 SF between 8 and 9 PM and 2 SFs between 12 am and 1 am. Similarly, for 2014 the largest number of SF occurred between 2 am and 3 am—the maximum number of SF events is 4 when the ionosonde is collecting data every 15 minutes. For 2008 there was at least one SF during each hour between 8 PM and 5 AM of the selected day.

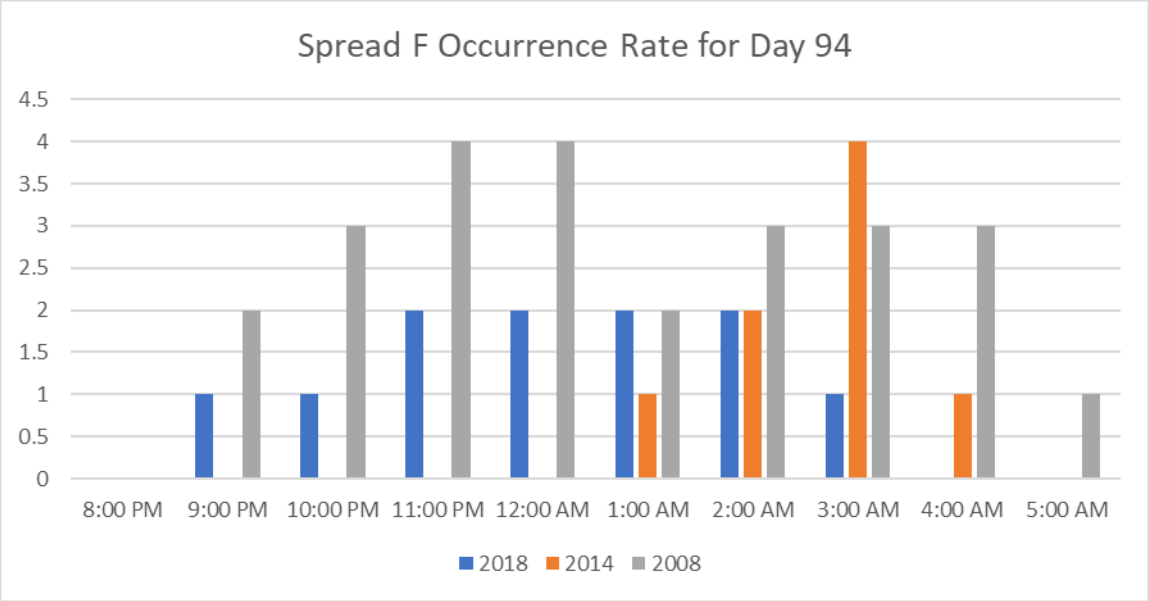

Figure 4: Number of SF events as function of time for same day (94) in the years 2008 (gray), 2014(orange) and 2018(blue) between 7 pm and 5 am local time in JRO.

MACHINE LEARNING METHODS

There are two key categories in terms of algorithms used for establishing the models: SVMs and CNNs. The algorithms within the CNNs approach can be further split into three subcategories, namely autoencoder, simple baseline neural network, and transfer learning. The following sections describe each algorithm, the data used for the algorithms, and the software programs used to implement the algorithms. Two key variables, the mathematical algorithms and the data, are carefully tested in the experiments described.

1. **Support Vector Machines (SVM)**

SVM refers to models that are defined by selecting a special set of vectors from the input space; the selected vectors are called support vectors. Let each input sample or pattern, such as an ionogram, be encoded as a n-dimensional vector X, where n corresponds to the number of features or attributes of the sample—in this case, number of pixels times number of color channels (red, blue, and green). For the classification task between SF and no SF, each sample is assigned a label. The samples with SF may be in Class A, and those without SF are in Class B. For SVM the class labels are 1 and -1, as this helps with the mathematical formulation of the problem.

The goal of SVM is to find a hyperplane (a hyperplane is a subspace with dimen-

sion = n-1, for example a line is a hyperplane of two-dimensional (n=2) space (Bridgelall, 2010) that separates the samples into distinct classes. Each sample can be thought of as a point in the input vector space. Samples that are in one class would then be points on one side of the hyperplane, and samples in the other class are points on the other side of the hyperplane. In SVM the goal is to find not just any hyperplane, but one that maximizes the minimum (perpendicular) distance from the hyperplane to a set of special points (which are actually samples) that are called *support vectors* (Shiri, 2004). These support vectors, a small subset of the input dataset, define how wide the margin of separation between the two classes is.
In order to define the optimal hyperplane the problem is framed as an optimization problem where the goal is to find the optimal value (in this case the *maximizing coefficients*) for the function L_d:

$$L_d \;=\; \sum_{i=1}^{i=m} a_i \; -\frac{1}{2} \sum_{i=1}^{i=m} \sum_{j=1}^{j=m} a_i \; a_j y_i y_j \; \left(x_i \bullet x_j\right) \qquad (1)$$

in the *original* input space. In this equation, a_i is the coefficient associated with the i^{th} sample, and $x_i \bullet x_j$ is the dot product between two input samples x_i and x_j with their respective labels y_i and y_j. The indices of summations are from 1 to the total number of samples, denoted as m.

When the input patterns are not linearly separable, as is the case for a complex problem such as the one developed here, a mapping $\Phi(x_i)$ to a higher dimensional space is required. In that high dimensional space, called feature space, the patterns can then be separated linearly. The goal thus becomes to optimize the following function, L_d,:

$$L_d \;=\; \sum_{i=1}^{m} a_i \; -\frac{1}{2} \sum_{i=1}^{i=m} \sum_{j=1}^{j=m} a_i \; a_j y_i y_j \; \left(\Phi(x_i) \bullet \Phi(x_j)\right) \qquad (2)$$

where $\Phi(x_i)$ is the vector corresponding to sample i in the (often much higher dimensional) feature space. Performing the dot product $\Phi(x_i) \bullet \Phi(x_j)$ in the feature space is computationally expensive. The idea of kernel functions was introduced to minimize this computational overhead. The kernel function $K\left(x_i,\; x_j\right)$ uses the low dimensional (original) input space vectors to obtain the same value that is obtained when doing the dot product in the higher dimensional space (Hearst, 1998). The optimization goal then becomes to find the optimizing coefficients for:

$$L_d \;=\; \sum_{i=1}^{m} a_i \; -\frac{1}{2} \sum_{i=1}^{i=m} \sum_{j=1}^{j=m} a_i \; a_j y_i y_j \; \mathrm{K}\left(x_i \bullet x_j\right) \quad (3)$$

In both cases the optimization problem is subject to the constraint that:

$$\sum_{i=1}^{m} a_i y_i = 0, \; a_i > 0 \quad (4)$$

Different kernel functions exist, and the trained classifier may perform better or worse depending on the choice of the kernel (Berwick, 2003). The coefficients associated with the *support vectors* are nonzero and determine the location of the optimal hyperplane. The rest of the coefficients do not contribute toward determining the location of the hyperplane. Which class a sample x_{new} belongs to depends on the sign of the value from the following sign function:

$$sign(w^T x_{\text{new}} + b) \quad (5)$$

where $w = \sum_{i=1}^{m} a_i y_i x_i$ and w^T is the transpose, noting that x_i is vector of features that define a sample. If the output of the sign function is negative the sample is said to be in one class (e.g., SF), and when the output is positive the sample is said to be in another class (e.g., Normal).

The implementation was done in MATLAB using the Support Vector Machine classification module. The module has a function fitcsvm that takes in the training samples which are represented as one large matrix X. A row of X is one training sample; and, each row has the number of columns corresponding to the number of features in the data. For the case discussed here the number of features is 1,260,000 because each ionogram image is 700 by 600 pixels by 3 color channels. (Pixels here refers to the logical pixels rather than physical pixels on some specific screen). Y is a column vector of labels. So, Y(137) is the label for the 137^{th} training sample. Y is either 1, to mean that the corresponding training sample has characteristics of SF; or Y can be 0, to mean that the corresponding training sample does not show sufficient signs of SF. The module has hyperparameters that are useful for the training. These hyperparementers include the different kernels, which in MATLAB include 'BoxConstraint', 'KernelScale', 'KernelFunction', 'PolynomialOrder', or 'Standardize'. When implementing, a specific portion of the total training set is held back for testing. Suppose Holdout = 0.15 then at each run MATLAB chooses, randomly, 15% of the data and excludes this portion in the training stage. Because the choice of which 15% is to be held back is random, the accuracy of the trained classifier varies.

Only the data from the year 2013 was used for creating the SVM model, since

this was primarily done to check the feasibility of the model. Several runs were performed while varying the hyperparameters for the model itself as well as for the training process. The accuracy was consistently not good, ranging between 50 % and 77 %. This method was thus abandoned. SVMs are known for being mathematically intuitive and are thus a good point of entry for this kind of classification problems. But given their poor performance, the next attempt explores convolutional neural networks, which, though harder to explain why they work, work, and in this particular case worked very well.

1. Convolutional Neural Networks

Convolutional neural networks (CNN) are a special neural network configuration that have demonstrated good learning ability when presented with input signals where neighboring elements form some semantic unit or feature; this is the case for images, and CNNs have thus been extensively used for image classification and detection tasks (Simonyan and Zisserman, 2015). Convolution is a mathematical operation between two matrices that outputs another matrix. To perform the convolution, a kernel (also called mask), which is a matrix of weights, operates on the input signal. The input signal in this case is the ionogram. The ionogram is represented as a 3-dimensional tensor of dimensions *height (600 pixels)*, *width (700 pixels)*, and *depth (3 color channels: Red, Blue, Green)*. To perform the convolution the kernel slides across the input image linearly combining the elements of the ionogram that are under the kernel; each element in the ionogram is multiplied by the corresponding element in the kernel and then the products are summed (Choi, Coyner, Kalpathy-Cramer, Chiang, & Peter Campbell, 2020), giving an output matrix. Properties of the kernel that need to be decided by the network designer include padding, stride, and kernel size. Padding is used to deal with two issues: to handle the pixels on the edge of the input signal and to make up for the fact that the size of the output matrix from the convolution operation is smaller than the original input matrix. A padding of 2 means that two new rows (often with zeros as the values) are added to the top and bottom of the input and two new columns are added to left and right of the input signal (ionogram). The values of the added pixels can be zero. Stride refers to the number of rows and columns to skip when moving the kernel across the image. Finally, kernel size (k pixels by r pixels) refers to the size of the kernel matrix. A small kernel looks at more local features, while a larger kernel looks at more global features. The actual values of the kernel matrix (these are called weights) are what the neural network learns during the training phase; this is the unique feature of CNNs in that one does not need to manually define the kernel weights.

A filter can be thought of as a collection of many kernels—a kernel is the 2D matrix that acts on data from one channel (which need not necessarily be color channel), whereas a filter combines these kernels into one 2D matrix via the element-wise addition. One also adds the bias to the output from the convolution filter. Finally, there is an activation function that maps the convolution output to a desirable range. A frequently used activation function is the recti-

fied unit function $f(x) = max(0, x)$, where x would be the value at each pixel of the convolution map obtained after the convolution operation (Stuchi, Boccato, & Attux, 2020).

CNNs require far fewer parameters than a regular fully-connected neural network. Therefore, training a CNN can be faster than a regular neural network. Even more interestingly is that the kernel weights are learned so that one does not need to manually determine what features to look for in an image—the CNN determines the abstract features that are important (Choi et al., 2020).

There are three key configurations within the CNN approach that have been explored in this work: an autoencoder architecture that enables learning with little supervision, a simple handcrafted CNN model, and the transfer learning approach which uses a more sophisticated network architecture that had been trained on public data in which there were no ionograms.

1. Autoencoder Architecture: An unsupervised approach to classifying SF

The autoencoder is a network architecture that uses CNN layers of different sizes and arranged in such a way that the output of the network has the same dimensions (or shape) as the input. Given an input X, the model must encode the input, and subsequently decode it in order to produce a reconstructed sample X_{est}. Then a comparison is done between the input and output to quantify the difference (error) between the input and the reconstructed. This reconstruction error is a measure of how well the model is able to encode the high-dimensional input samples into a low-dimensional space, with smaller error signifying better ability at encoding.
When the autoencoder concept is applied to this two-class problem of SF and normal, the key idea is to use a training dataset with a lot more normal samples than SF samples. When the autoencoder model is taught to learn to reconstruct ionograms in this dataset, the model will learn very well how to reconstruct ionograms from the class that has more samples, which in this case is the normal class. However, when provided with a sample from the less frequently seen class the autoencoder makes a greater error in reconstruction (Borghesi, Bartolini, Lombardi, Milano, & Benini, 2019). A threshold on the reconstruction error is set; if the error is above the chosen threshold the sample is classified as SF. The advantage of this approach is that only those samples used for testing purposes are required to be labeled. The approach is therefore classified as unsupervised learning since there is no need to provide the labels for training.
The actual implementation was carried out using the Keras Application Programming Interface (API) (Francois, 2015) within the Tensorflow framework. The model consisted of the layers shown in Table 1.

Table 1: Layers of the autoencoder, showing an architecture where the input shape is the same as the output shape.

Layer (type)	Output Shape	Number of Parameters
Input_1 (InputLayer)	[(None, 400,300,3)	0
Encoder (Functional)	(None, 380)	45601868
Decoder(Functional)	(None, 400,300,3)	45723795

In the autoencoder structure shown in Table 1 the input of shape 400 pixels by 300 pixels by 3 channels is mapped to a low-dimensional space (of dimension only 380). This process encodes the input. The latter part of the neural structure is the decoder; it decodes the encoded sample and attempts to recreate the original image. When the model has learnt the key features of the input space well enough, it recreates the original image with greater accuracy (Borghesi et al., 2019). By setting a threshold on the amount of error made in recreating the original image, one can classify the test input as either Spread F or not.

In this architecture, the original ionograms are rescaled from 700 pixels wide by 600 pixels high down to 400 pixels high by 300 pixels wide. The scaled-down input then goes through several layers (not shown in table) until it is encoded in a 380-dimension space. This is the end of the encoder section of the auto-encoder. From the encoder layer, the goal is to reconstruct the input. The final layer has the dimension of 400 pixels by 300 pixels by 3 channels just as the input. Thus, the mathematical comparison can be done between the input and the decoded ionogram.

Data from all years described in Section 2 are used for this algorithm. In total, there are 40232 ionograms used as the training set. Another 9896 ionograms were reserved for *testing* and thus were manually classified into either the Normal or the SF class. The threshold for error set is on the *test set*, where the error is the mean of the squares of the difference between the original input and the reconstructed input. The error threshold of 0.11 was ultimately chosen after the training; when the reconstruction error is above 0.11 then the sample is classified as SF. If the error is less than 0.11 then the sample is classified as Normal. It was expected that samples *without* SF would result in less error since the dataset was established with more Normal samples.

Metrics used to evaluate the ML model results are defined in the Section 4. From this autoencoder method the following results were obtained: precision of 63 percent, recall of 69 percent and accuracy of 63 percent. These results show that the method not provide much success compared with previous studies.

Thus, we further developed other ML model architectures as explained in the following sections. A key difference of the following approaches from the Autoencoder is the use of labelled data. Therefore, they fall under the paradigm of supervised learning.

1. **Simple Model Architecture**

This is a supervised learning approach, where the images are labelled prior to

their use in the model. The model is given the ionograms and the associated label of SF or Normal. The initial model is architecturally quite simple with few layers as well as few neurons per layer; this model is much faster to train (than the models used in transfer learning models explained below). This simple model has the structure shown in Figure 5.

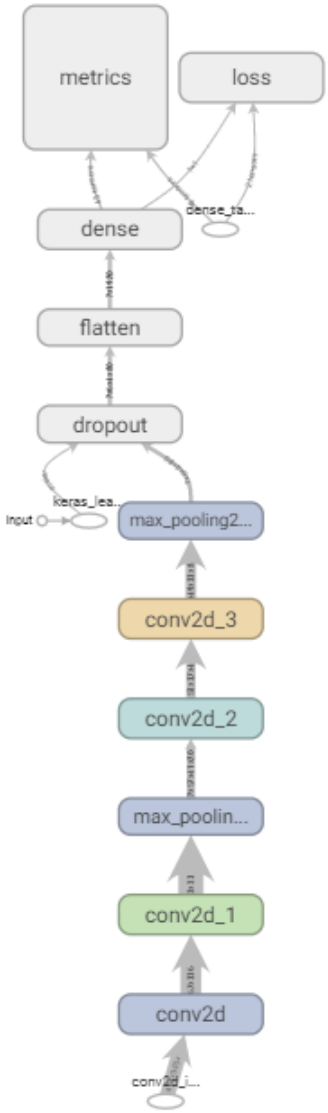

Figure 5: Simple neural network architecture used for training SF classifier show-

ing the different layers. Conv2d means 2D convolutional layers, max_pooling2 is a maxpooling layer, dropout deactivates some connections between the two layers, and flatten converts multidimensional input to a vector. The metrics are used to define the goodness of the model while the loss is the function that the one tries to optimize.

Figure 5 shows the simple base model with only 7 kernel layers (excluding pooling layers and dropout layers). Note that the weights of the kernels are what the model learns. Thus, if there are many kernel layers there would be a larger number of weights that need to be calculated. The *Conv2d* layers are convolutional layers and *max_pooling* are max pooling layers which perform downsampling. The flatten layer is used to unwrap the matrices from earlier layers into a vector whose elements can then be connected to a dense layer. Finally, the metrics and loss function values are calculated. The goal in training the model (machine) is to reduce the loss as the training progresses. An overview of the shape at each layer is shown in the Table 2 below.

Table 2: The different layers of the simple architecture that is the baseline for comparison with transfer learning models

Layer (type)	Output Shape	Number of p
Con2d (Conv2D)	(None, 296,296,10)	760
Conv2d_1(Conv2D)	(None, 292,292,20)	5020
max_pooling2d (MaxPooling2D)	(None, 36, 36, 20)	
conv2d_2 (Conv2D)	(None, 32, 32, 40)	20040
conv2d_3 (Conv2D)	(None, 28, 28, 80)	80080
max_pooling2d_1 (MaxPooling2D)	(None, 3, 3, 80)	Not applicab
dropout (Dropout)	(None, 3, 3, 80)	Not applicab
flatten (Flatten)	(None, 720)	Not applicab
dense (Dense)	(None, 1)	721
Total parameters	Sum of all parameters	**106,621**
Trainable parameters	Parameters whose values are calculated during training	106,621

The goal of the training is to find the values for the 106,621 parameters which minimize the cost function. The implementation was through Keras API. While the model was easy to train, the model learning metrics did not improve with respect to epoch number. It was therefore decided to look into transfer learning. The primary reason for starting with the simple model was to have a basis for comparison; this model is otherwise not discussed further.

1. **Transfer Learning: Using Community Architectures**

Transfer learning starts with existing architectures that were trained on data that did not include ionograms and then the pretrained model is finetuned with the ionogram images. This has been shown to be an effective way to train a model based on a small dataset (Weiss, Khoshgoftaar, & Wang, 2016).

The original model architectures are well-designed and subsequently trained on very large datasets, they have features that can be useful in other classification problems. This is so because the earlier layers of a neural network can act as feature extractors (Stuchi et al., 2020) and can thus recognize patterns that exist across many images, for example edges and contrasts. Edges exist in images of human faces, cars, and ionograms. Thus, the same model can detect this feature in the many possible scenarios.

The following architectures were tested in this study: VGG16, InceptionV3, and ResNet50. These community models were slightly modified and finetuned for the task of classifying SF. The training was carried out using Graphics Processing Units through the Google Colaboratory Platform and those provided in-house at Machine Learning and Data Analytics at Nanyang Technological University Singapore under the Electrical and Electronic Engineering Department.

VGG16

The VGG16 network architecture was introduced in year 2014. The 16 in VGG stands for 16 weight layers. It was trained on over one million images divided across 1000 classes. In 2014 it achieved 92.7% top-5 test accuracy on the ImageNet dataset (Jia Deng et al., 2009). This architecture has convolutional layers that use the rectified linear unit function for introducing nonlinearity. At the top it has two fully connected layers with each of the two layers having 4096 neurons. Then the output is wired to another fully connected with 1000 neurons (Simonyan & Zisserman, 2015). The 1000 is because VGG19 was being trained to classify inputs into the 1000 classes which are part of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Finally, this fully connected layer connects to a softmax layer. At the softmax layer, each node or neuron i applies the following function to its input z:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (6)$$

The result is that only one of the 1000 neurons will have the highest output; in fact, the sum of all the outputs has to be equal to 1. Since each node represents one class, then when node k has the highest value (that is, $\sigma(z)_k$ is max) the interpretation is that the network has predicted the input as belonging to class k. For a more concrete example, suppose there is fully connected layer with 7 neurons, representing 7 classes, then when softmax function is applied the result may be:

$$outFCAfterSoftmax = [0.00,\ 0.00,\ 0.63,\ 0.19,\ 0.14,\ 0.025,\ 0.012] \quad (7)$$

This identifies the third output as highest. One of the reasons for this softmax layer mapping is to allow interpretation as probability. It can be said that there

is a 63 percent chance that the input belongs to class number 3, and only a 1.2 percent chance of the input belonging to class 7. Similarly, if there were only 2 classes (that is, the softmax layer would have 2 neurons) then the output might be:

$$outFCAfterSoftmax = [0.90,\ 0.10] \quad (8)$$

This indicates that the sample belongs to class 1 because, in equation (8) 0.90 is bigger than 0.10.

Specifically for the two-class problem, the softmax layer can instead be replaced by a sigmoid layer with just one neuron, but with similar functionality. With only two classes—SF and normal—as opposed to the 1000 classes for which VGG19 was designed, the top layer (softmax) with the 1000 neurons can be replaced with a sigmoid-based layer. The sigmoid function, specifically the logistic function, is:

$$\sigma(z)_i = \frac{e^{z_i}}{1+e^{z_i}} \quad (9)$$

This function is at the very last layer. When the output is greater than 0.5, the input is said to belong to class A, which is SF. When the output of the sigmoid function is less than 0.5 then the sample ionogram is said to belong to the class of Normal ionograms.

InceptionV3

First presented in year 2015, InceptionV3 is designed to have inception modules. Each module in the series may consist of applying the following operations to the input of the module (Szegedy et al., 2015):

1. 1x1 convolution filters
2. 3x3 filter for the medium features spread across few pixels
3. 5x5 filter for detecting slightly more global features. A large filter size is computationally more expensive. One of the tricks used is to do the same operation—that is, convolution with a 5 by 5 filter—but using a block that is made up of two layers. The two are a 3 by 3 pixel convolution layer, followed by another 3 by 3 layer that takes the output from the previous 3 by 3 (directly without applying an activation function).
4. Maxpooling, an operation that extracts the maximum value from an input region

This module can then be used as one structural unit. For ease of computation the modules use dimension reduction so that 3x3 and 5x5 filters are computed

after applying 1x1 filters (Szegedy et al., 2015); this design allows for a deep model but with computational efficiency. The same minor architectural modifications, as for VGG16, were made to InceptionV3: remove top layer, add new layer with 1024 units, dropout layer with 0.05 ratio and a sigmoid layer. The training is also done on GPUs.

ResNet50

It had been recognized that deeper networks are better able to do classification. However, it was also seen that when the network depth crosses some threshold it is no longer beneficial to make it deeper; in fact, the network's performance degrades. ResNet was created to aim for a deep network architecture that does not sacrifice performance (He, Zhang, Ren, & Sun, 2016). A key feature of the ResNet50 architecture is that of skipping some layers in a proper way.

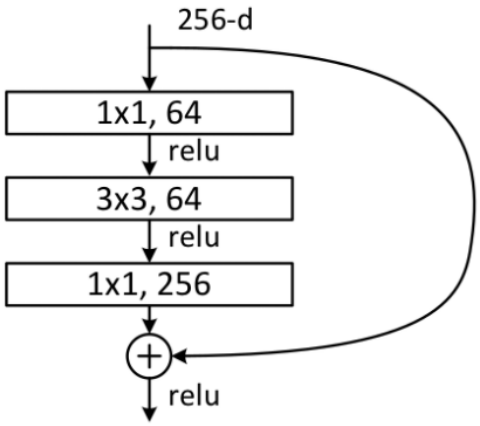

Figure 6: A residual module for the ResNet50 architecture. Many such modules are connected together to form the full model.

In the schematic shown in Figure 6, the input (256-d) is going into the module that consists of three layers. The first layer has 64 different kernels of size 1 by 1. This layer operates directly on the input. The output from this layer goes through a nonlinear function rectified linear unit (relu). The output from the relu operation then is the input to the next layer. The next layer consists of kernels of size 3 pixels by 3 pixels; as in the first layer, there are also 64 such kernels. The output is again nonlinearly mapped using relu whose output is the input to the next layer. This next layer has 256 kernels each of size 1 pixel by 1 pixel. Note that all kernel sizes are implicitly of shape n by m by d where d is the number of channels—where the term channel as used here does *not* refer to the red, blue and green color channels. So, if the input is an image of 224 by 224 by 3 (i.e., 224 pixels wide by 224 pixels high by 3 channels) then a 3 by 3 pixel kernel that operates on this image implies that its kernel size is actually 3 by 3 by 3.

For the ResNet50 architecture, the input goes through this module made of three layers. But instead of simply passing the output of the three layers to the following set of layers, ResNet adds the original, unmodified input to the output of the three layers (He et al., 2016). This is a distinguishing feature of ResNet and its variants. Once the output from the three layers is summed with the original input, the relu function can be applied. The output from this *relu* operation is passed to the next set of layers and so on. In this case the output is passed to another residual block, consisting of three layers with similar architecture. This led to three residual blocks of the same kind being concatenated. The output from the three residual blocks are then passed to another *type* of residual block and so on. In the end the ResNet50 architecture has 50 kernel layers, with four types of residual blocks.

In order to train the models the Keras API was also used. The pretrained model had been trained for 1000 classes. The modification made was to remove the top layer which is used for 1000-class classification and replace it with a sigmoid layer. However, before the sigmoid layer, two new layers were added: one fully-connected layer of 1024 units/neurons and a dropout layer. The dropout layer has a ratio of 0.05; this means that during the training process this dropout layer deactivates (that is, sets to 0), at random, 0.05 or 5 percent of the weights connecting this layer to the previous layer. Table 3 shows the overall changes made for this model.

Table 3: Changes made to the ResNet50 architecture to customize for the task of identifying SF.

Parameter	**Value**
Pretrained Model	ResNet50
Optimizer	tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07, amsgrad=False)
Neurons at first added dense layer	
Retrained layers from original	
Dropout fraction for dropout layer	
Input shape	by 300
Datasets	Training: Y2008,09,10,11,12,13,14,16,19 (20611) Validation: 2017, 2015 (5716 samples) Testing: 2018 (2050 samples)

The parameters in Table 3 are defined for the training phase, for this architecture and similarly for the other architectures. The value for the Pretrained Model can be “ResNet50”, “InceptionV3” or “VGG16”. The optimizer is the

algorithm that is used for actually finding the weights that minimize the loss. The concept of optimizer algorithm is similar to how the Newton-Raphson algorithm can be used to find zeros/roots of a polynomial, but instead of finding zeros the optimizer algorithm helps in finding approximate weight values that minimize the loss or cost function. The cost function quantifies the difference between desired output and the output predicted by the model at each training step. The API comes with several options for the optimizing algorithm including "Adagrad", "RMSProp", and "SGD". One also can choose to tune the optimizing algorithm itself—for example in the third row of Table 3, the hyperparameters learning_rate, beta_1, beta_2, epsilon, and amsgrad are specifically for configuring this Adam optimizer. It has been shown that the choice of the optimizing algorithm can affect the final model, but this variable was kept constant in this work.

4. RESULTS AND SUMMARY

The dataset for training and validation ionograms has 26327 samples of which 15770 have SF and 10557 are normal. To get the validation dataset the ionogram samples were randomly split: 24 percent for validation and the rest for training. This resulted in 20010 samples for training and 6317 for validation. All three transfer learning methods (VGG16, InceptionV3, and ResNet50) are adopted with the same ionogram data to create three different ML models.

When training and later comparing models, the parameters in Table 3 are kept the same across models. Several data augment techniques are implemented in ionograms prior to ingesting them in the neural network. First, the original image size of is rescaled before feeding into these ML methods. The original input images are 700 pixels wide and 600 pixels high. However, most algorithms for use in Transfer Learning require smaller input shape. The transfer learning base models used here are VGG19, InceptionV3, and ResNet50, which respectively have input tensor shapes of 224 by 224, 299 by 299 and 224 by 224. However, rather than scaling each ionogram dataset to its own input shape, all ionograms were scaled to the same shape of 300 pixels by 300 pixels irrespective of the base model. A sample ionogram that has been scaled down and used as an input to the neutral network is shown in Figure 7. The key information about the ionogram is still being captured despite the rescaling. The advantage in rescaling is that the number of parameters whose values need to be learned are significantly reduced, which results in faster training and better learning.

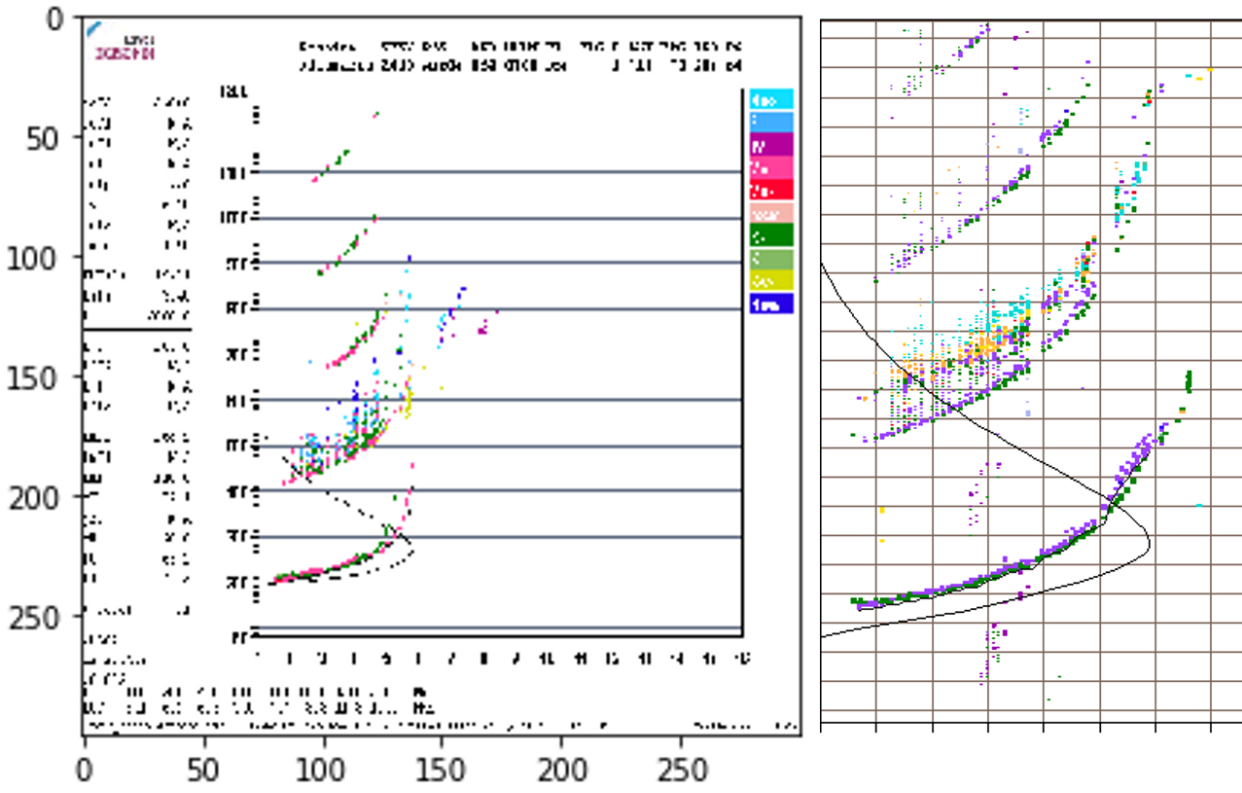

Figure 7: The full ionogram of 700 pixels wide by 600 pixels wide is rescaled to 300 pixels by 300 pixels as shown in the blurred ionogram on the left, prior to sending as input into the network. This blurred version is what the neural networks "sees". The image on the right shows a cropped ionogram, to exclude information that is not expected to be relevant for identifying SF; a collection of such cropped images is used in one of the experiments.

The other modification to the input that was tested was to crop out the extra information that is on ionograms that is auto-scaled by a software such as ARTIST and useful for the human but does not seem necessarily relevant for the SF detection. A set consisting of the same number of samples as the original dataset but with the ionograms cropped was created; a sample is shown at the right of Figure 7. Then a model was trained with these cropped ionograms to check whether the model that results is better at detecting SF.

The metrics used in this work are defined as follows, where positive refers to an ionogram with SF:

1. Accuracy of the model with respect to the training and validation set. Accuracy is a measure of how many samples are

correctly put into their expected class.

2. False Negatives—This is the number of samples that are incorrectly identified as not having SF when they in fact do have SF.

3. False Positives—This is the number of samples that are incorrectly identified as having SF when they do not have SF.

4. True Positives—This is the number of samples that are correctly identified as having SF.

5. True Negative—This is the number of samples that are correctly identified as not having SF.

6. Area Under the Curve (AUC), where the curve is a plot of False Positive Rate (x) vs. True Positive Rate (y). True Positive Rate = True Positives / (True Positives + False Negatives) while False Positive Rate = False Positives / (False Positives + True Negatives) (Brownlee, 2020)

7. Precision—This is a measure of how many of those identified as SF are indeed SF. Precision = True Positive/(True Positive + False Positive). Suppose the model identifies 11 samples as having SF. Upon inspection it turns out that only 7 of those 11 are actually SF. Then the precision is $7/11 = 0.63$

8. Recall—This is a measure of how many of the *actual* SF's are identified. Recall=True Positive/(True Positive + False Negative) For example, if it is known that there are 10 SF's and the algorithm is able to identify 6 of them then the model recall is $6/10 = 0.6$.

9. Loss—the loss (also called cost) function for which the training procedure seeks the minimum. The one used here is the binary cross entropy. Ideally the loss function should be decreasing as the training proceeds. This is because in training a model the goal is to find the set of parameters/weights that minimizes the value of the cost function.

During the training and validation phases, these nine metrics were tracked and used as the basis for deciding which is the better model (Brownlee, 2020). Figure 8 demonstrate the Precision throughout the training based on three different ML models. Values of Precision suggest that the model based on the ResNet50 architecture does the most learning, improving its precision from 0.6 at the beginning of training (first epoch) to 0.87 at the 120^{th} epoch. An epoch is one round of training in which all samples will have been seen by the model.

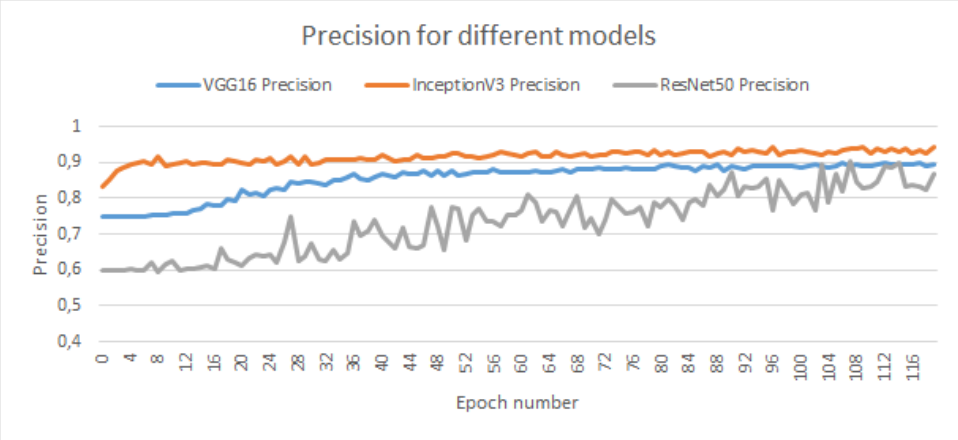

Figure 8: Precision for the three models, VGG16 (blue), InceptionV3(orange) and ResNet50(gray), with respect to epoch number.

Figure 9 also shows the AUC metric, which combines precision and recall. The ResNet50 architecture again shows the greatest amount of learning, from 58 percent at epoch 0 to about 90 percent by the 120^{th} epoch. Across other metrics, ResNet50 performed well, too. It was thus chosen for further experimentation.

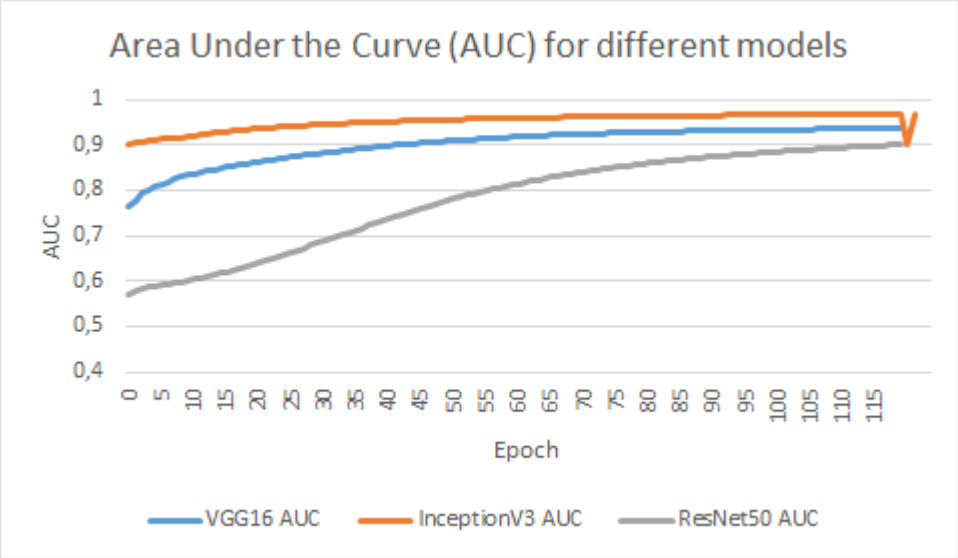

Figure 9: Area Under the Curve (AUC) for the three models, VGG16 (blue), InceptionV3 (orange) and ResNet50 (gray), with respect to epoch number.

Since the metric values showed that the best model is the one based on the ResNet50 architecture, two more experiments, namely cropping and removal of 5-minute cadence data, are further implemented to investigate if these changes

in the input images impact the model performance. The ionograms normally contain extra information that are irrelevant to the echoes, such as $foF2$ and $foF1$ on the left side of an ionogram; such information is automatically generated by the autoscaling software (e.g., ARTIST). Since this information is not useful for the training, cropped ionograms are tested and one example is demonstrated in Figure 8. To evaluate impact of different time-cadence of ionograms on the ML models, removing the 5-minute cadence data is also conducted.

Figure 10 shows that the model based on cropped data starts out with a higher precision (0.75) than the model based on the full size (non-cropped) ionograms, which starts with a precision of 0.60. The model based on the full ionograms shows much more significant learning such that at the 120^{th} epoch its precision is 0.86 while that from the cropped data is around 0.86. Based on this, using cropped ionograms does not offer much advantage. Results from the case with images with 15-minute cadence show that the precision starts from 0.74 and increases to around 0.88 at the 120^{th} epoch. While this is slightly better than the model based on the full ionograms, it does not seem like a significant enough difference to justify the preprocessing necessary to isolate 5-min cadence from 15-minute ones. However, from a physics point of view it may be worth further exploring the cause for the difference.

[CHART]

Figure 10: The precision of the ResNet50-based model using the fullsize dataset (blue), cropped data (orange), and dataset that excludes 5-minute cadence ionograms (gray).

Before these experiments, it is expected that a much more complex model such as ResNet50 would perform better than a simpler model—though this is not always guaranteed. This study also demonstrates that VGG16 and InsepctionV3 did not perform well on identifying the SF event. Since all these models had been established with images that did not contain ionograms it may be that their great skill at distinguishing features among the images they had seen comes at some cost; they may lack the ability to generalize. There were no ionograms in the ImageNet dataset which had been used to train the ResNet50, VGG16 and InceptionV3 models. Transfer learning works best when there are similarities between the original dataset and the new smaller dataset—that is, the smaller the distance of samples from the source images, the more effective the transfer learning (Azizpour, Razavian, Sullivan, Maki, & Carlsson, 2015). Additionally, network parameters such as width and depth as well as the choice of the optimization algorithm seem to affect the efficacy of the transfer learning (Azizpour et al., 2015). These factors may explain why some of the models used here did not perform well on the task of SF prediction.

The test set used for this work consists of ionograms from 2018 with 2050 ionograms. There are 1342 SF samples and 708 Normal samples. Using the full ionograms and the ResNet50 architecture, without retraining pretrained layers, the values for the metrics are listed in Table 4. Out of the 1342 SF samples the

trained model can correctly identify 1170 samples (true positives). Similarly, out of 708 Normal samples the algorithm can correctly classify 649 samples (true negatives). There are only 59 samples which are incorrectly classified as having Spread F when they do not show SF (false positive).

Table 4: Metric results based on the test set of 2050 samples for the ResNet50-based model only.

Metric	**Description**	**Value**
True Positives (TP)	Correctly identified SF case	1170
False Positives (FP)	SF case identified as Normal	59
True Negatives (TN)	Normal case identified correctly	649
False Negatives (FN)	SF case identified as Normal	172
Accuracy		0.89
Precision	TP/(TP+FP)	0.95
Recall	TP/(TP+FN)	0.87
Area Under the Curve	Area under the precision-recall curve	0.96

Assuming that the metrics are normally (Gaussian) distributed, confidence intervals can be obtained (Hazra, 2017)

$$interval = z * \left(\frac{accuracy*(1-accuracy)}{n}\right)^{\frac{1}{2}} \quad (10)$$

Where n is the number of samples and z is the value from the standard normal distribution corresponding to the desired confidence interval. There are n=2050 samples in the test set.

For a 95% significance level $z = 1.96$

$$interval = 1.96 * \left(\frac{0.89*(1-0.89)}{2050}\right)^{\frac{1}{2}} = 0.014 \quad (11)$$

This can be interpreted as saying that at 95% significance level the "true" accuracy of the model is 0.89±0.014 (uncertainty = 0.014), or between 0.88 and 0.90.

To summarize our results, several approaches of ML techniques were carried out in this study. One approach used SVM while the other is based on CNNs. The accuracy of the SVM model was between 55% and 77%. The first attempt of CNNs based on autoencoder the accuracy hovered around 63%. Finally with transfer learning an existing well-designed architecture was used as the starting point. The model was then modified by removing the top layer then adding

a new fully-connected layer, a dropout layer and a sigmoid layer. Performing these same modifications across ResNet50, InceptionV3, and VGG16 showed that ResNet50 is the best based on Area Under the Curve, Precision and Recall metrics.

CONCLUSION AND FUTURE WORK

In this study, a total of 28377 ionograms primarily around equinoxes are labelled. There were 17112 samples with SF events and 11265 normal samples from 2008-2019. The filenames and the corresponding label for each ionogram are available to the community and can be accessed at this link: https://doi.org/10.21979/N9/BHGVGH.

Two types of ML approaches are adopted to create an automatic algorithm for identifying the SF event from these ionograms: SVM and CNN. For the CNN architectures, three methods are further explored: autoencoder, a simple architecture, and transfer learning. While the autoencoder model required very little labelling, it turned out to not be as effective, with accuracy less than 63 percent. Similarly, while the simpler model architecture is much faster to train, it did not show sufficient learning skill and the accuracy remains around 88 percent from the first epoch to the 120^{th}. Through experiments from this study, the transfer learning with multiple community architectures has demonstrated success in solving this problem. The model based on the ResNet50 architecture appears to be the most effective model with the AUC improving from 0.64 to 0.86 in 120 epochs. Our study further investigates the impact of various information in these images on the results. Results show that the model trained with the cropped images do not show much improvements based on these evaluation metrics. The model with only 15-minute cadence images also has no significant impact on the overall performance. Therefore, one can simply use the full ionograms with the ResNet50 base architecture and with the architectural modifications made in this study (i.e., remove top layer and replace with a dense layer of 1024 neurons, add a dropout layer with rate of 0.05, and a sigmoid classification layer) to achieve the same results.

The next steps of this study will include creating a web app that streams real-time ionograms from JRO in order to automatically label the real-time images and to establish a catalog of SF event for future use. This ML technique significantly reduce the need for manual process in identifying the onset of an event and is an extremely important step for exploring SF forecasting based on time-series ionograms in the near future.

Acknowledgment

We are grateful for the Singapore International Graduate Award that C. Luwanga is a beneficiary of. T.-W. Fang was supported by NSF SWQU

grant (AGS 2028032) and NOAA Space Weather Prediction Center. Y.-J. Lee was funded under NSF grant (AGS 1552017). We also thank Justin Mabie, at the University of Colorado Boulder, for the help in interpreting and working with ionograms. The data used in this work can be found here: https://doi.org/10.21979/N9/BHGVGH hosted by NTU Singapore Data Repository (DR-NTU) to whom we're grateful. Public access to experiments: https://ui.neptune.ai/luwangac/SFDetection/experiments?viewId=standard-view.

References

Alfonsi, L., Spogli, L., Pezzopane, M., Romano, V., Zuccheretti, E., De Franceschi, G., … Ezquer, R. G. (2013). Comparative analysis of spread-F signature and GPS scintillation occurrences at Tucumán, Argentina. *Journal of Geophysical Research: Space Physics*, *118*(7), 4483–4502. https://doi.org/10.1002/jgra.50378Azizpour, H., Razavian, A. S., Sullivan, J., Maki, A., & Carlsson, S. (2015). From generic to specific deep representations for visual recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2015-Octob*, 36–45. https://doi.org/10.1109/CVPRW.2015.7301270Berwick, R. (2003). *An Idiot's Guide to Support vector machines (SVMs): A New Generation of Learning Algorithms Key Ideas*. 1–28. Retrieved from http://www.cs.ucf.edu/courses/cap6412/fall2009/papers/Berwick2003.pdfBhaneja, P., Earle, G. D., Bishop, R. L., Bullett, T. W., Mabie, J., & Redmon, R. (2009). A statistical study of midlatitude spread F at Wallops Island, Virginia. *Journal of Geophysical Research: Space Physics*, *114*(4), 1–9. https://doi.org/10.1029/2008JA013212Bora, S. (2017). Ionosphere and radio communication. *Resonance*, *22*(2), 123–133. https://doi.org/10.1007/s12045-017-0443-8Borghesi, A., Bartolini, A., Lombardi, M., Milano, M., & Benini, L. (2019). Anomaly detection using autoencoders in high performance computing systems. *CEUR Workshop Proceedings*, *2495*, 24–32. https://doi.org/10.1609/aaai.v33i01.33019428Bridgelall, R. (2010). *Introduction to Support Vector Machines*. *2011*(January 1st), 1. Retrieved from https://www.ugpti.org/smartse/resources/downloads/support-vector-machines.pdfCander, L. R. (2019). *Ionospheric Space Weather*. https://doi.org/10.1007/978-3-319-99331-7_9Chapagain, N. P., Fejer, B. G., & Chau, J. L. (2009). Climatology of postsunset equatorial spread F over Jicamarca. *Journal of Geophysical Research: Space Physics*, *114*(7), 1–7. https://doi.org/10.1029/2008JA013911Choi, R. Y., Coyner, A. S., Kalpathy-Cramer, J., Chiang, M. F., & Peter Campbell, J. (2020). Introduction to machine learning, neural networks, and deep learning. *Translational Vision Science and Technology*, *9*(2), 1–12. https://doi.org/10.1167/tvst.9.2.14Floer, M. (2020). *Design and I mplementation of a Software De fi ned I onosonde A contribution to the development of distributed arrays of small instruments*. (June).Hazra, A. (2017). Using the confidence interval confidently. *Journal of Thoracic Disease*, Vol. 9, pp. 4125–4130. https://doi.org/10.21037/jtd.2017.09.14He, K., Zhang, X., Ren, S., & Sun, J.

(2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 770–778. https://doi.org/10.1109/CVPR.2016.90Hearst, M. (1998). *Support Vector Machines*. 339–359. https://doi.org/10.1007/978-3-030-40189-4_7Jia Deng, Wei Dong, Socher, R., Li-Jia Li, Kai Li, & Li Fei-Fei. (2009). *ImageNet: A large-scale hierarchical image database*. 248–255. https://doi.org/10.1109/cvprw.2009.5206848Jiang, C., Yang, G., Liu, J., Yokoyama, T., Komolmis, T., Song, H., … Zhao, Z. (2016). Ionosonde observations of daytime spread F at low latitudes. *Journal of Geophysical Research: Space Physics*, *121*(12), 12,093-12,103. https://doi.org/10.1002/2016JA023123Kalita, B. R., Nath, S. J., Bhuyan, P. K., Khandare, A., & Kulkarni, A. (2019). SAMEERDU—Digital Ionosonde: Brief System Description and Initial Results from a Low-Latitude Location Dibrugarh. *Radio Science*, *54*(11), 1142–1155. https://doi.org/10.1029/2019RS006813Kintner, P. M., Humphreys, T., & Hinks, J. (2009). GNSS and ionospheric scintillation. *Inside GNSS*, *4*(4), 22–30. Retrieved from http://www.insidegnss.com/auto/julyaug09-kintner.pdfLan, T., Zhang, Y., Jiang, C., Yang, G., & Zhao, Z. (2018). Automatic identification of Spread F using decision trees. *Journal of Atmospheric and Solar-Terrestrial Physics*, *179*(April), 389–395. https://doi.org/10.1016/j.jastp.2018.09.007Li, G., Ning, B., Otsuka, Y., Abdu, M. A., Abadi, P., Liu, Z., … Wan, W. (2021). Challenges to Equatorial Plasma Bubble and Ionospheric Scintillation Short-Term Forecasting and Future Aspects in East and Southeast Asia. In *Surveys in Geophysics* (Vol. 42). https://doi.org/10.1007/s10712-020-09613-5Liang, J., Donovan, E., Nishimura, Y., Yang, B., Spanswick, E., Asamura, K., … Redmon, R. (2015). *Journal of Geophysical Research : Space Physics*. 1–24. https://doi.org/10.1002/2015JA021094.ReceivedMaruyama, T. (2002). Ionosphere and Thermosphere: Ionospheric Irregularities. *Journal of the Communications Research Laboratory*, *49*(3), 163–179.Milla, M. A. (2017). *The Jicamarca Radio Observatory : Instrumentation , Experiments , and Database Jicamarca Radio Observatory - Introduction*.Petrova, E., Podladchikova, T., Veronig, A. M., Lemmens, S., Bastida Virgili, B., & Flohrer, T. (2021). Medium-term Predictions of F10.7 and F30 cm Solar Radio Flux with the Adaptive Kalman Filter. *The Astrophysical Journal Supplement Series*, Vol. 254, p. 9. https://doi.org/10.3847/1538-4365/abef6dPillat, V. G., Fagundes, P. R., & Guimarães, L. N. F. (2015). Automatically identification of Equatorial Spread-F occurrence on ionograms. *Journal of Atmospheric and Solar-Terrestrial Physics*, *135*, 118–125. https://doi.org/10.1016/j.jastp.2015.10.015Reinisch, B. W. (2009). *THE DIGITAL PORTABLE SOUNDER DPS*. Lowell Digisonde International.Sahai, Y., Fagundes, P. R., Abalde, J. R., Pimenta, A. A., Bittencourt, J. A., Otsuka, Y., & Rios, V. H. (2004). Generation of large-scale equatorial F-region plasma depletions during low range spread-F season. *Annales Geophysicae*, *22*(1), 15–23. https://doi.org/10.5194/angeo-22-15-2004Shi, J. K., Wang, G. J., Reinisch, B. W., Shang, S. P., Wang, X., Zherebotsov, G., & Potekhin, A. (2011). Relationship between strong range spread F and ionospheric scintillations observed in Hainan from 2003

to 2007. *Journal of Geophysical Research: Space Physics*, *116*(8), 1–5. https://doi.org/10.1029/2011JA016806Shiri, A. (2004). Introduction to Modern Information Retrieval (2nd edition). *Library Review*, *53*(9), 462–463. https://doi.org/10.1108/00242530410565256Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–14.Stuchi, J. A., Boccato, L., & Attux, R. (2020). Frequency learning for image classification. *ArXiv*.Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594Weiss, K., Khoshgoftaar, T. M., & Wang, D. D. (2016). A survey of transfer learning. In *Journal of Big Data* (Vol. 3). https://doi.org/10.1186/s40537-016-0043-6Wiesemann, K. (2013). A short introduction to plasma physics. *CAS-CERN Accelerator School: Ion Sources - Proceedings*, (1), 85–122.

Figures and Captions

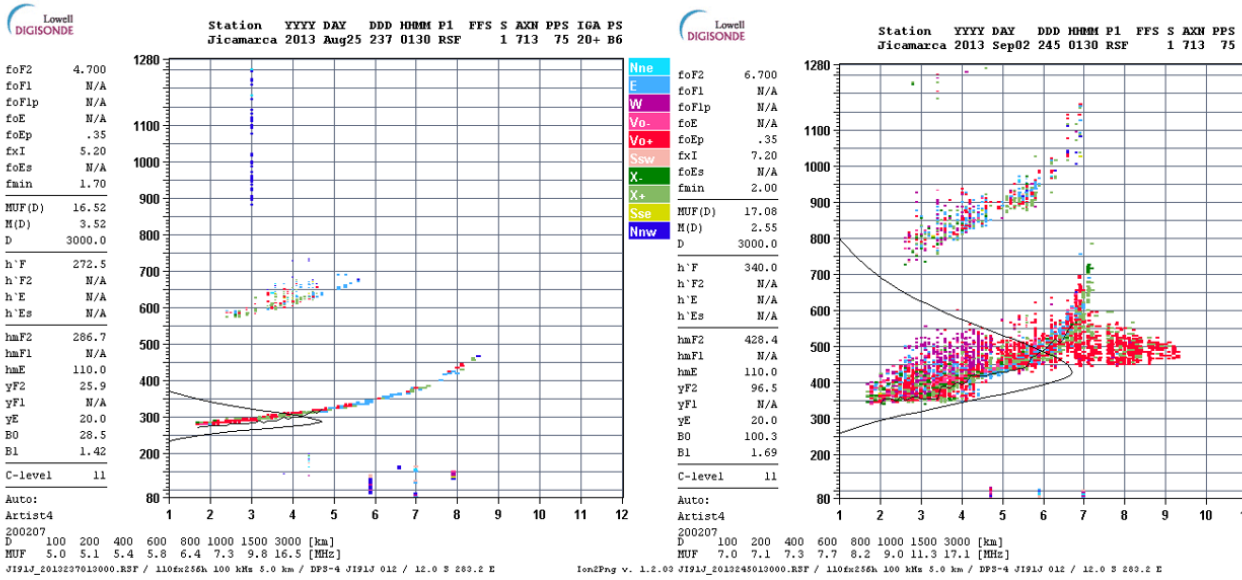

Figure 1: Ionogram without SF on the left and ionogram with SF on the right. The colors encode the following parameters: the direction from which the echo comes, the Doppler, and the mode of the echo—that is, whether it is an extraordinary mode or ordinary mode.

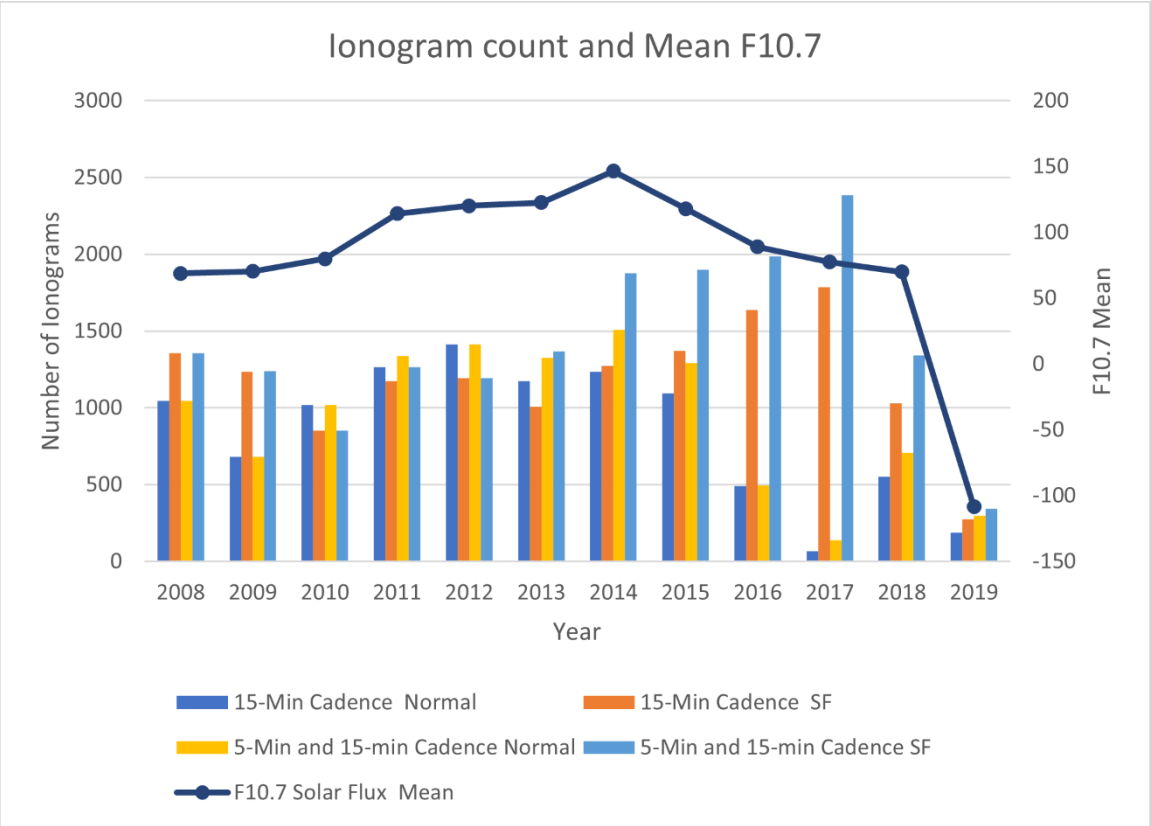

Figure 2: Number of ionograms from the SF class and Normal class as function of year. The set of ionograms of 15-minute cadence excludes ionograms collected every five minutes, while the 15-min and 5-min cadence set includes both the ionograms collected every 5 minutes as well as those collected every 15 minutes. The F10.7 index shows the phase of the solar cycle.

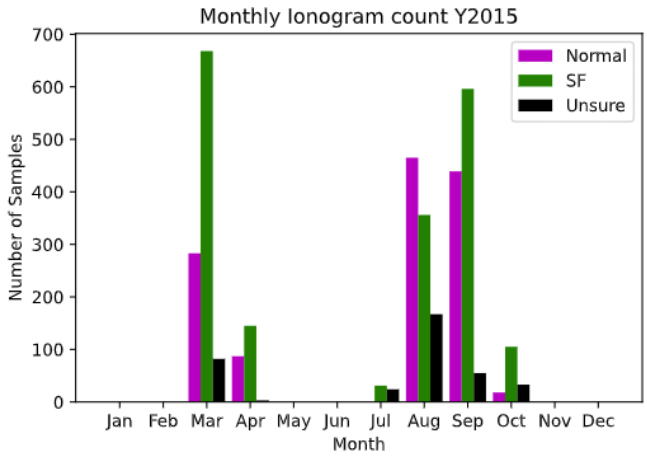

Figure 3: Monthly distribution of samples in year 2015 as divided into Normal (purple), SF (green) and Unsure (black). Unsure are not used in creating the models.

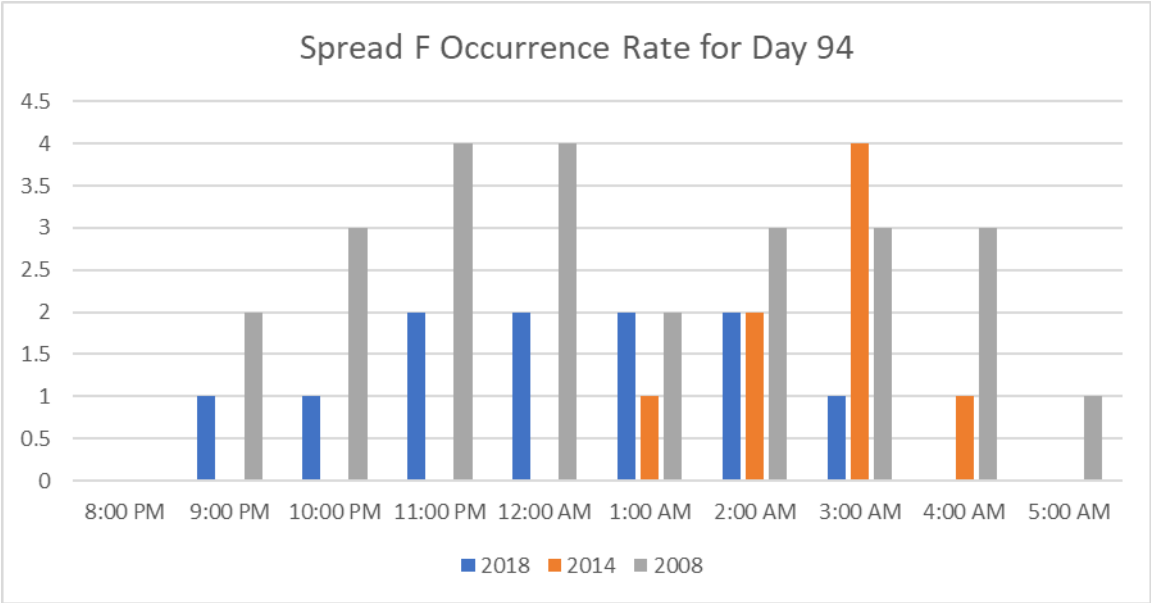

Figure 4: Number of SF events as function of time for same day (94) in the years 2008 (gray), 2014(orange) and 2018(blue) between 7 pm and 5 am local time in Jicamarca.

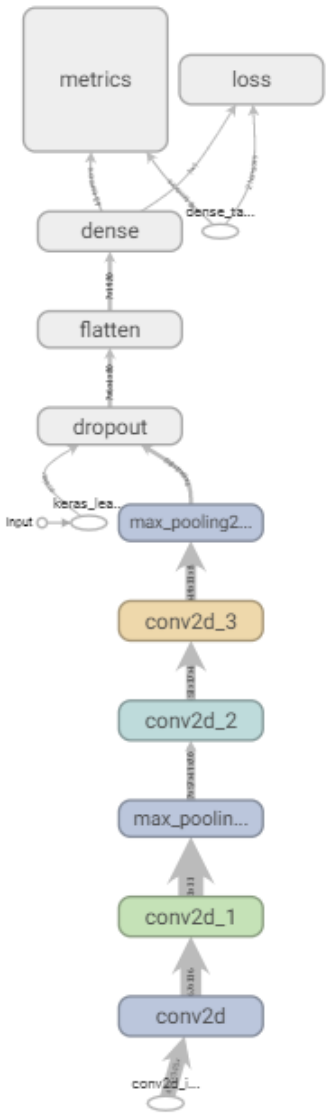

Figure 5: Simple neural network architecture used for training SF classifier showing the different layers. Conv2d means 2D convolutional layers, max_pooling2 is a maxpooling layer, dropout deactivates some connections between the two layers, and flatten converts multidimensional input to a vector. The metrics are used to define the goodness of the model while the loss is the function that the one tries to optimize.

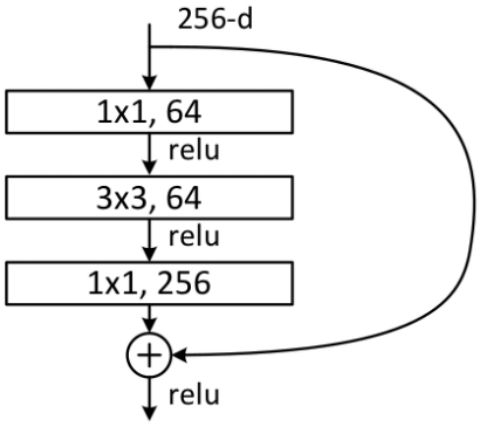

Figure 6: A residual module for the ResNet50 architecture. Many such modules are connected together to form the full model.

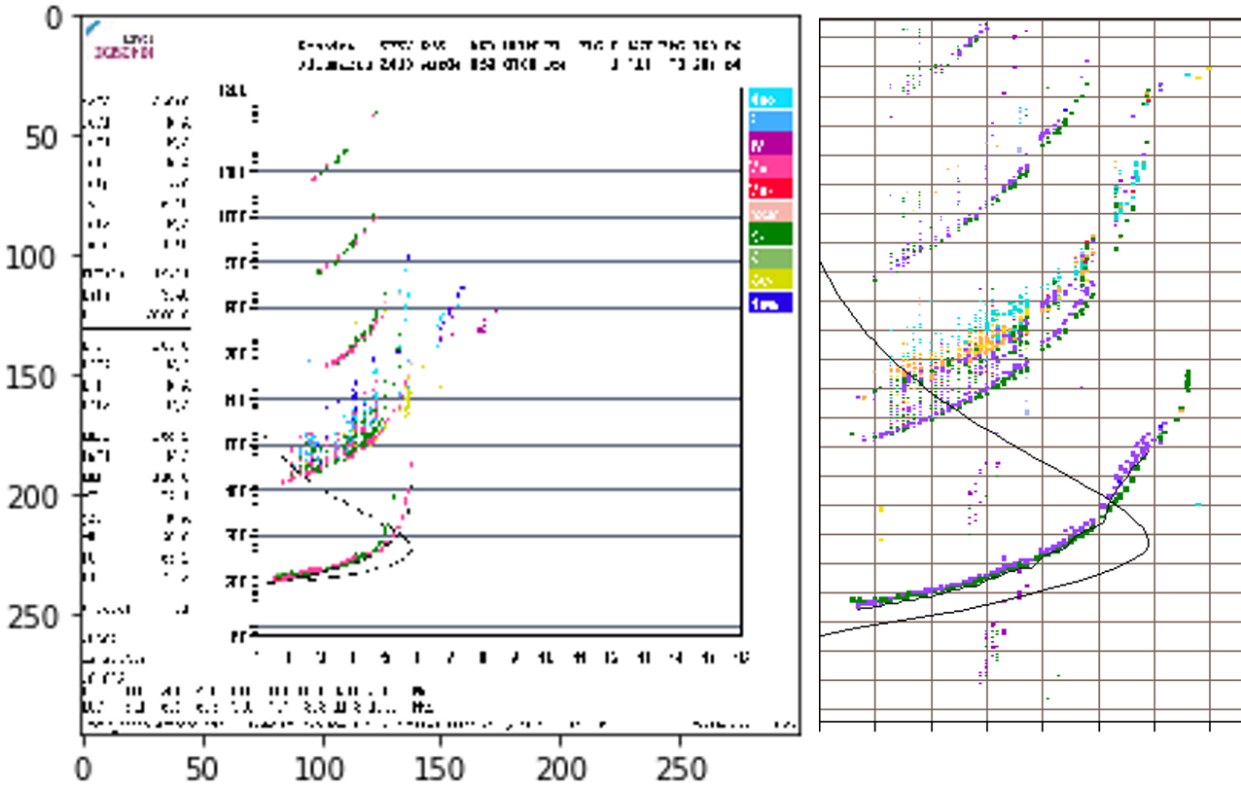

Figure 7: The full ionogram of 700 pixels wide by 600 pixels wide is rescaled to 300 pixels by 300 pixels as shown in the blurred ionogram on the left, prior

to sending as input into the network. This blurred version is what the neural networks "sees". The image on the right shows a cropped ionogram, to exclude information that is not expected to be relevant for identifying SF; a collection of such cropped images is used in one of the experiments.

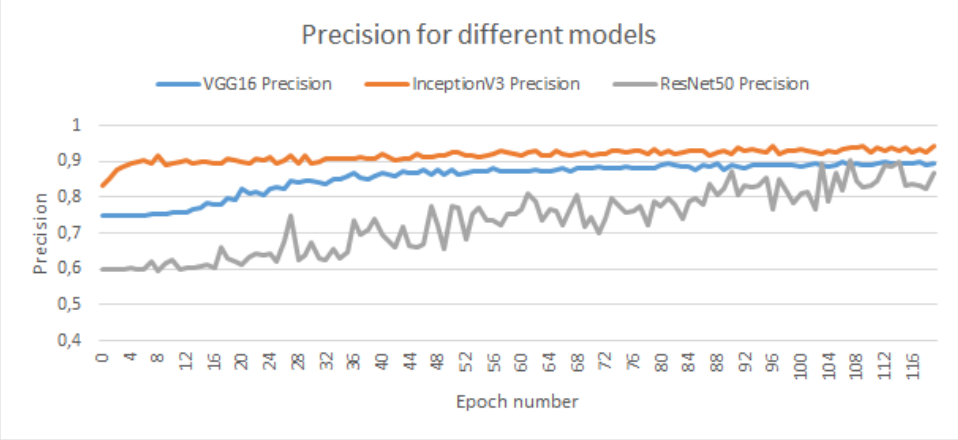

Figure 8: Precision for the three models, VGG16 (blue), InceptionV3(orange) and ResNet50(gray), with respect to epoch number.

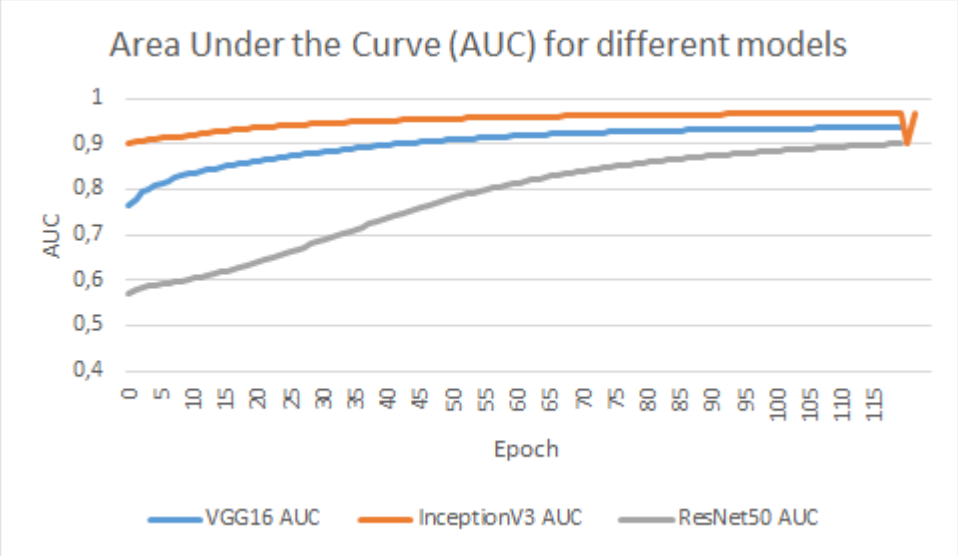

Figure 9: Area Under the Curve (AUC) for the three models, VGG16 (blue), InceptionV3 (orange) and ResNet50 (gray), with respect to epoch number.

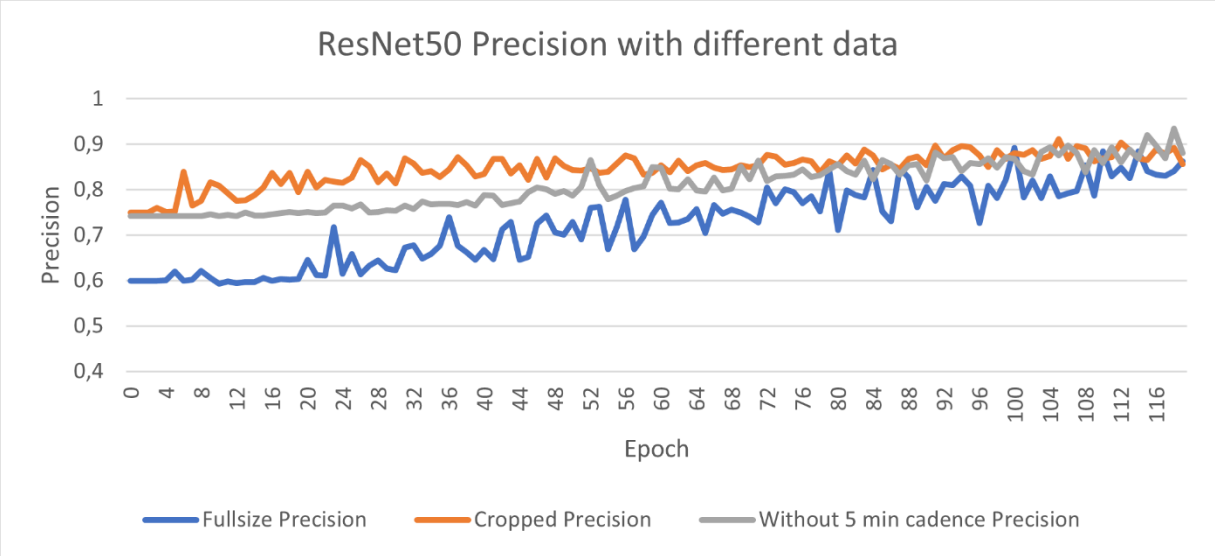

Figure 10: The precision of the ResNet50-based model using the fullsize dataset (blue), cropped data (orange), and dataset that excludes 5-minute cadence ionograms (gray).

Tables and captions

Table 1: Layers of the autoencoder, showing an architecture where the input shape is the same as the output shape.

Layer (type)	Output Shape	Number of Parameters
Input_1 (InputLayer)	[(None, 400,300,3)	0
Encoder (Functional)	(None, 380)	45601868
Decoder(Functional)	(None, 400,300,3)	45723795

Table 2: The different layers of the simple architecture that is the baseline for comparison with transfer learning models

Layer (type)	Output Shape	Number of p
Con2d (Conv2D)	(None, 296,296,10)	760
Conv2d_1(Conv2D)	(None, 292,292,20)	5020
max_pooling2d (MaxPooling2D)	(None, 36, 36, 20)	
conv2d_2 (Conv2D)	(None, 32, 32, 40)	20040
conv2d_3 (Conv2D)	(None, 28, 28, 80)	80080
max_pooling2d_1 (MaxPooling2D)	(None, 3, 3, 80)	Not applicab
dropout (Dropout)	(None, 3, 3, 80)	Not applicab

Layer (type)	Output Shape	Number of p
flatten (Flatten)	(None, 720)	Not applicab
dense (Dense)	(None, 1)	721
Total parameters	Sum of all parameters	**106,621**
Trainable parameters	Parameters whose values are calculated during training	106,621

Table 3: Changes made to the ResNet50 architecture to customize for the task of identifying SF.

Parameter	**Value**
Pretrained Model	ResNet50
Optimizer	tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07, amsgrad=False)
Neurons at first added dense layer	
Retrained layers from original	
Dropout fraction for dropout layer	
Input shape	by 300
Datasets	Training: Y2008,09,10,11,12,13,14,16,19 (20611) Validation: 2017, 2015 (5716 samples) Testing: 2018 (2050 samples)

Table 4: Metric results based on the test set of 2050 samples for the ResNet50-based model only.

Metric	**Description**	**Value**
True Positives (TP)	Correctly identified SF case	1170
False Positives (FP)	SF case identified as Normal	59
True Negatives (TN)	Normal case identified correctly	649
False Negatives (FN)	SF case identified as Normal	172
Accuracy		0.89
Precision	TP/(TP+FP)	0.95
Recall	TP/(TP+FN)	0.87
Area Under the Curve	Area under the precision-recall curve	0.96