Investigating the Use Of Convolutional Long Short-Term Memory Encoder Decoder Models in Conflict Prediction

Gareth Lomax

August 2019

1 Administrative Information

1.1 Candidate Information

Email Address: gcl15@ic.ac.uk
Github Alias: Garethlomax
Github: https://github.com/msc-acse/acse-9-independent-research-project-Garethlomax

1.2 Project Supervisor (External)

Dr Nils W. Metternich
Associate Professor in International Relations
School of Public Policy
Dept Of Political Science UCL
Email: n.metternich@ucl.ac.uk

1.3 Supervisors (Internal)

Prof Sanjeev Gupta
Professor of Earth Science
Faculty of Engineering, Department of Earth Science Engineering
Imperial College London
Email: s.gupta@imperial.ac.uk
Dr Steven Banham
Research Associate
Faculty of Engineering, Department of Earth Science Engineering
Imperial College London
Email: s.banham@imperial.ac.uk

Contents

Abstract

conflictlstm is a lightweight package facilitating the use of convolutional long short term memory (ConvLSTM) models to predict the occurrence of fatal conflict events in Africa at a PRIO grid cell level. The package allows easy construction, training and analysis of encoder decoder ConvLSTM models. The package also allows easy construction of conflict image sequence datasets. The package was used to produce predictions of conflict occurrence in Africa between $2012-01-01$ and $2014-01-01$. The best model produces predictions outperforming the $F1$ score and AUPR of the predictions of the state of the art VIEWS project, despite training on only 6% of the available features used by other efforts.

2 Introduction

Civil, interstate and terrorist conflict holds great influence in dictating the future of a country Collier and Hoeffler [2000].

Over the last century conflict has directly caused the death of millions, the destruction of cultures and societies, and has rendered parts of the earth virtually uninhabitable. Alarmingly recent studies have identified climate change as a conflict driving factor Hegre et al. [2017] Weidmann and Schutte [2017].

In addition to this analyses have demonstrated that the traditional role of 'policy experts' in guiding governmental decisions can have adverse consequences, due to their poor predictive abilities.Tetlock [2018].

The advent of large scale dissagregated datasets has allowed large scale analyses of conflicts, and the production of successful conflict forecasting models Tollefsen et al. [2012a] Hegre and Sambanis [2006] Croicu and Sundberg [2017] Bormann et al. [2018]. As policy decisions are increasingly made based on model predictions, it is vital that a robust range of conflict prediction tools are produced in order to assist NGOs and at risk groups with strategic decisions, with the eventual goal of pre-empting and reducing the damage caused by potential conflicts though an early warning system. This need is greater than ever in the face of a global rise in polarisation, populism, and the exacerbating effects of climate change.

While a number of conflict prediction systems exist, such as the ICEWS project run by the United States Defense Advance Research Projects Agency (DARPA) O'Brien [2010], few large scale conflict forecasting efforts that openly broadcast their predictions to the public currently exist Hegre et al. [2019a].

In practice conflict predictions usually consists of a binary variable for each spatio-temporal unit of analysis denoting whether one or more battle related deaths can be attributed to a violent conflict in a specified region in space and time. Current efforts in conflict prediction are centered around theoretical models, agent based modelling and statistical machine learning approaches. The most promising efforts have used ensembles of machine learning models trained on large dissagregated data sets of geo-located economic and socio-political predictors Hegre et al. [2019a]. Primarily the machine learning approaches have centered around variants of time lagged logistic regression and random forests Muchlinski et al. [2016], due to their effectiveness in a wide range of problems, and the inherent class imbalance present in conflict data.

Numerous efforts in the conflict prediction community have explored the phenomena of conflict contagion and diffusion Metternich et al. [2017]. Conflict has been shown to have a strong spatial dependency as a result of these diffusive effects. Conflict has also been shown to be dependant on predictor variables which themselves have strong spatial dependencies, such as the movement of different ethnic groups, mountainous region percentages, weather, and drought distributions Hegre and Sambanis [2006].

In contrast to this, the vast majority of prediction efforts do not use methods that fully leverage the spatial structure of the available data. It is often common practice to engineer features using spatial lags, which do not fully capture conflict structures and force isotropy into the conflict model Croicu and Eck [2018].

This project seeks to explore the efficacy of deeper machine learning techniques when applied to conflict data. Convolutional Long Short-Term Memory (ConvLSTM) Shi et al. [2015] models have produced best in class results in a number of spatio-temporal prediction tasks such as weather nowcasting Shi et al. [2015]. We hypothesize that utilising techniques which explicitly extract information from the spatial structure of conflict data will result in increased prediction accuracy of the spatio-temporal development of conflicts. We also hypothesize that exploiting the spatial structure of the data will provide a better level of model performance per predictor variable than current models.

The project detailed in this report was focused on the production of a python package to enable the use of ConvLSTM models in conflict forecasting. The package allows users to easily define multi-GPU Pytorch implementations of encoder - decoder ConvLSTMs, as well as utilise the building blocks behind this functionality to define their own models. The package also implements methods to construct image-based datasets out of existing conflict data, to allow for further exploration of image techniques in conflict forecasting in addition to methods to display and analyse the conflict predictions produced.
Using the functionality detailed above ConvLSTM encoder-decoders were trained on a new image sequence dataset of conflict clusters. When the produced model is trained on conflict data with significant spatial distribution (> 2.7 conflict events per month) we outperform the state of the art model when comparing the Area Under Precision Recall graph (AUPR) and the F1 score. We observe Area Under the Receiver Operator Characteristic graph (AUROC) scores comparable to that of the state of the art. We explore training methods and find a significant advantage using modified class weighted Binary Cross Entropy to produce significantly improved results. In line with the surrounding literature we observe deeper LSTM models perform well in comparison to smaller models, although significantly increase model overhead. These results were produced using approximately 6% of the features used by the state of the art models, clearly demonstrating the efficacy of the ConvLSTM models.

This project represents the first use of image analysis and convolution techniques to predict conflict of which we are aware. Our model produces performance metrics which match or outperform those of the state of the art VIEWS project Hegre et al. [2019a], despite using a much smaller selection of (poorly performing) prediction features from the same dataset used by the VIEWS project. The project demonstrates that convolutional Long Term Short Memory models are a powerful tool in conflict prediction.

An overview of both conflict, and sequence prediction is provided in section 3. A Software Requirement Specification (SRS) was produced to outline the required functionality required to test the hypotheses in section 2 and is shown in section 4.1. Section 4.2.1 details the production ethos and methodology of our software. The dependencies and requirements of the software are detailed in section **??**. The results produced using the implemented software are presented in section 6 and discussed in section 7.

3 Literature Review

3.1 Conflict

3.1.1 Terminology

Note that from here we use the UCDP definition of conflict which defines a conflict as a contested incompatibility between two parties which results in at least 25 battle related deaths per year Croicu and Sundberg [2016]. We will also discuss various data that have been identified as having relevant to conflict prediction. This data is usually identified as relevant from theoretical or agent based studies targeting specific conflict causation systems e.g imbalances in wealth distribution between different ethnic groups in a region Mitra and Ray [2014]. Once identified in the literature these data are then typically used as input data into machine learning algorithms in subsequent studies. In line with the machine learning community we will refer to these data as predictors. Predictors usually describe the physical and social geography of a region in space, as well as the economic climate Hegre and Sambanis [2006]. They may be categorical data, such as the binary presence of oil wells in the region they describe, or continuous, such as the nighttime luminescence of a region, as measured by satellite Tollefsen et al. [2012a].

3.1.2 Overview

Broadly we may separate conflict prediction efforts into two categories; Model driven and Data driven. Model driven approaches seek to impose theoretical or semi-theoretical rule based frameworks to predict future conflict usually in the form of agent based modelling, whereas data drive approaches leverage machine learning methods, such as random forests or logistic regression Muchlinski et al. [2016]. Machine learning approaches are advantageous when seeking overall predictive power, however they often lack transparency, and are less successful when modelling factors surrounding specific causes of conflict. Hegre et al. [2017]It should be noted however that many of the apriori theoretical models are still informed by observation of the data by subject experts. Theoretical and agent based models usually center about investigating specific conflict causation systems, and often much of the choice of predictors in data driven efforts is based on results from strict agent based models which have highlighted the roles of specific conflict predictors, such as the exclusion of different ethnic groups or climate change Sarsons [2015] Weidmann [2011]. Models use a variety of spatial representations to predict conflict. Originally predictions focused on large scale units of analysis, using a country by country representation, using country wide aggregated predictor variables, such as GDP, to predict the presence of conflict in the country as a whole. Numerous studies have noted the advantage of disagregating data into smaller geographic representations, such as Weidmann and Ward [2010], which represents Bosnia using sub-national municipalities, and forecasts the probability of violence in each municipality, or Hegre et al. [2019a] which uses a continent wide Cartesian grid representation to forecast conflict on a cell grid level. Other works such as Guo et al. [2016] have represented countries as city networks to predict conflict due to the ease of representing both geographic and political connections using single or multi layer networks. They use predictor variables corresponding to city entities rather than predictors dependent on geographic regions.

3.1.3 Conflict Predictors

A wide array of relevant systems have been identified as contributing to conflict. Collier and Hoeffler [2000] identifies low economic growth as a significant cause of conflict due to its reductive effect on a states ability to stamp out protests and violence. This conclusion is furthered in Fearon and Laitin [2003], in which they authors demonstrate that poor economic progress raises the opportunity cost involved with rebellion and contesting civil wars and that poor economic conditions favour rebel recruitment. Mitra and Ray [2014] Uses econometric theory applied to Hindu-Muslim violence in India and examines the effect of raising incomes of either of the groups, concluding that an income shock is likely to lead to grievance and violence between the groups. Alongside this Olsson [2007] finds that the abundance of precious resources typically leads to lower economic progress and an increased susceptibility to conflict. Rainfall variation is often used as a proxy to study the effect of income shocks on conflict. Bohlken and Sergenti [2010] have used this to find correlation between the prevalence of rioting and rainfall. Sarsons [2015] expands on this and finds that rainfall still acts well as a predictor of unrest even in areas

adjacent to dams, which should be immune to sudden droughts, suggesting that climate has a larger role to play. In Hegre and Sambanis [2006] the authors conduct sensitivity analyses on 88 variables commonly used in conflict prediction with the objective of giving a reality check in a field containing a large amount of contradictory findings. They again support the idea of the strong effect of GDP on conflict likelihood, demonstrating that a 1 standard deviation reduction of income across a region raises the probability of civil war by 65 percent. They also show that regions containing mountainous and rough terrain can also predict the presence of violence well, in agreement with Fearon and Laitin [2003], a study which reasons that the natural hideouts favour insurgent groups hiding there. Alongside this they also demonstrate that large militaries prevent intrastate conflict and recent political instability is the most robust predictor for civil war. The presence of civil war in adjacent countries is also noted as a robust predictor, as would be expected given the large range of work on conflict diffusion and contagion.

3.1.4 Spatial Dependence

Clear amongst each of these works is the importance of the relative position of competing groups in conflict prediction. The effect of position of conflict actors is of huge importance in conflict prediction, both due to many conflict predictors being a function of location (avg annual temperature, presence of oil resources, susceptibility to climate change ect) and which are often clustered**?**, but also due to the phenomena of conflict contagion and diffusion. Weidmann and Ward [2010] Compares two models of conflict applied to geo-located conflict data from the Bosnian war (March 1992 - October 1995) and demonstrates the improvement in out of sample prediction of violence prediction (binary: presence vs absence) of 3% accuracy by including nearest neighbours spatial information and demonstrates the tendency of conflict to spill over between neighbouring groups. Metternich et al. [2017] Furthers this line of enquiry and demonstrates contagion between different conflict actors as being reliant on both of the sending and receiving groups utilising an bi-linear network model, and demonstrate that the effect that the ethnic exclusion of both the sending and receiving conflict groups is dependant on the geographic proximity of the two groups. Guo et al. [2016] Use a geographic city network based model and show that the betweenness centrality of a city can explain above 80% of the variance in terrorist attacks in city location, and use an agent based model to demonstrate this is as a result of the high likelihood of fuzzy cultural boundaries at high betweeness centrality locations.

3.1.5 Data

The project will use event and predictor data from the PRIO grid and UCDP-GED data sets Tollefsen et al. [2012a] Sundberg and Melander [2013] . The PRIO (Peace Research Institute of Oslo) data set is widely used in conflict research to provide predictor information at a sub national level. The PRIO grid consists of a vector cell grid of dimensions (0.5 x 0.5) degrees of latitude and longitude covering the globe. Each cell contains variables sorted into conflict relevant groups (socioeconomic, resources and climate.) The UCDP data set (Uppsala conflict data program) encodes the number of deaths due to a conflict on a monthly basis. For a given political situation to be classed as a conflict, it must meet a threshold of 25 battle related fatalities per year. For a conflict to be recorded all groups involved in the conflict must be clearly defined. Croicu and Sundberg [2016]. A version of the UCDP base has been adapted to fit the PRIO grid structure, and covers the years 1989-2018. The UCDP encodes the number of deaths at a given grid cell at each month. The norm in conflict prediction is to reduce this to a binary presence variable, where efforts are focused on the prediction of presence of violence, rather than its magnitude. We will shape our prediction efforts around this for ease of comparison with other efforts. We note that due to the complexities involved with extracting data from conflict zones, the data used will require spatial imputation in approximately 5% of cases.

3.2 Sequence Prediction

Conflict prediction, in essence, is a spatio-temporal sequence prediction problem. As has been noted above in section 3.1.3, conflict is highly dependant on spatial configurations. As a result we find it prudent to seek the use of methods that are reliant on spatial encoding and have produced outstanding results in spatio-temporal prediction tasks.

3.2.1 Recurrent Neural Nets

Recurrent Neural Nets (RNNs) and their variants are the preferred machine learning architecture for temporal problems. The RNN is an extension of a standard feed forward neural net designed to handle temporally related sequences of input data, which may be of variable length. Graves [2013] This is achieved by the introduction of a recurrent hidden layer h_t whose activation is a weighted combination of the previous hidden layer h_{t-1} and the input at the current time step x_t. This differentiates RNNs from feed forward neural networks: the hidden layer h can be said to be self connected. Graves [2012] This may be seen visualised in figure 1. Note that it is common practice to represent the RNN in an "unfolded state" to visualise the output process, and is especially useful in deep RNN networks. RNNs take a sequence of inputs $\mathbf{x}$ and map to a sequence of outputs $\mathbf{y}$:

$$\begin{aligned} h_t &= \mathcal{H}(W_{ih}x_t + W_{hh}h_{t-1} + b_h) \\ y_t &= W_{hy}h_t + b_y \end{aligned} \tag{1}$$

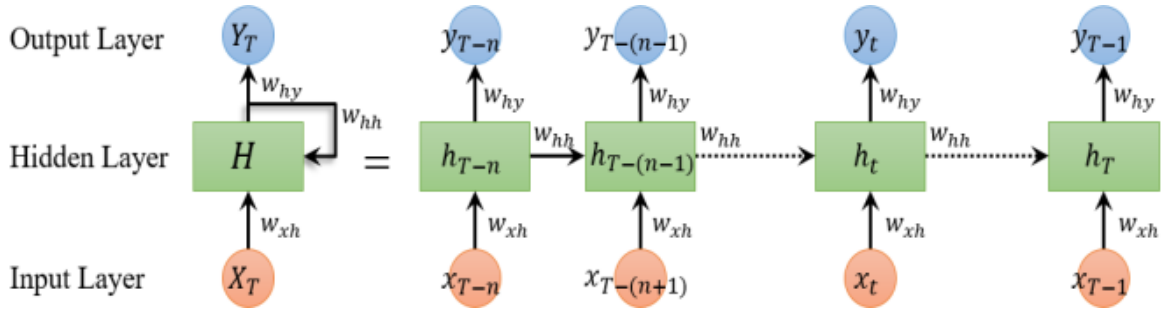


Figure 1: Diagram of folded and unfolded RNN. At timestep t the RNN takes an input vector x_t which combined with the the hidden layer from the previous timestep, h_{t-1} and a bias term, b_h according to an activation function Graves [2013]. The output a given timestep y_t is a weighted fully connected transformation of the hidden state. The incorporation of the fully connected layer, modulated by W_{hh} allows sequence information pass between different sequence steps, allowing the RNN to recognise patterns present in the data. The RNN is represented in its "unfolded" format on the right, demonstrating how the hidden state passes through the system as time progresses. Image source: Cui et al. [2018]

Where h_t is the hidden layer, y_t is the output, x_t is the input, W_{ih}, W_{hh} and W_{hy} are the weight matrices for the fully connected layers and b_h and b_y are biases. $\mathcal{H}$ is the hidden layer activation function, which in practice is usually a sigmoid. Graves [2012] Graves [2013] RNNs perform well on sequence based tasks due to residual information from preceding inputs remaining in the network. In simplistic terms; they remember the previous inputs and may draw inferences based on the sequence as a whole. Unfortunately basic RNNs are unreliable when applied to extracting long temporal relationships in the data due to issues with gradients vanishing and exploding whilst training via back propagation, as has been demonstrated in Hochreiter [2003]. As a result of this flaw training with back propagated gradient descent models is often unfeasible or impossible. Several additions to vanilla RNN architectures have been introduced to counteract this flaw. Aside from efforts to produce training methods immune to the gradient problems. The Long Short-Term Memory architecture (LSTM) was proposed in 1997 in **?**, although the method went relatively unexplored for a number of years, before being used to produce an array of best in class results. In the time between several additions to the LSTM structure have been proposed and accepted by the community. Gated Recurrent Units (GRUs)Chung et al. [2014] were also introduced in 2014 and have been involved in healthy competition with LSTMs with both producing impressive results. Both LSTMs and GRUs introduce gated memory approaches to the activation functions of the RNN hidden layer to overcome the vanishing gradient issue. We choose to focus on LSTMs due to the greater amount one may tailor their activation functions. Here we define the LSTM:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + w_{co} \circ c_t + b_o) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
\tag{2}
$$

Where f_t, i_t, $o_t$and c_t are the forget gates, input gates, output gates and cell memories respectively. σ is the sigmoid activation function. In this case $a \circ b$ denotes element wise multiplication. As we can see from the above equations the role of the memory cell is significant. It enables longer term memory storage as the cell memory is selectively updated, instead of updating at every time step as in a vanilla RNN. As a result it retains significant features for longer. In other words, this allows information to skip multiple time steps if the information is relevant. The FC-LSTM is a trivial extension of the LSTM where the x_t and h_t layers are extended to fully connected vectors, allowing large amounts of interconnected information to travel through the LSTM at each time step. This is also useful as it allows categorical information to be used in LSTMs by utilising one-hot encoding Rodríguez et al. [2018]. FC-LSTMs have performed very well in temporally dependant tasks, including in natural language processing (NLP) and sequence prediction. Notable results include Vinyals et al. [2015] in which the state of the art BLEU-1 captioning score was raised from 25 to 59. While FC-LSTMs are highly performant in sequence to sequence (seq2seq) tasks involving suitably linear data such as streams of encoded words, a number of works have explored methods using LSTMs to process spatially related data. Convolutional methods are the clear state of the art when extracting information from gridded data sources, especially image data Sharma et al. [2018]. Approaches such as those documented in Srivastava et al. [2015] use pretrained convolutional neural nets (CNNs) to encode image data at each sequence step, taking representations from either the networks high level percepts or directly from the final fully connected layer and passing this to the FC-LSTM. This approach has been explored in both supervised and unsupervised fashions to produce good results in learning video representations. They also explore the use of LSTM for future prediction using an implementation of the encoder-decoder system first proposed in Sutskever et al. [2014]. While this works well at the input to state stage as the spatial relationships are captured by the CNN input into the LSTM, the FC-LSTM does not preserve the spatial encoding in the state to state transitions due to its fully connected nature. I.e once the spatial data has been extracted

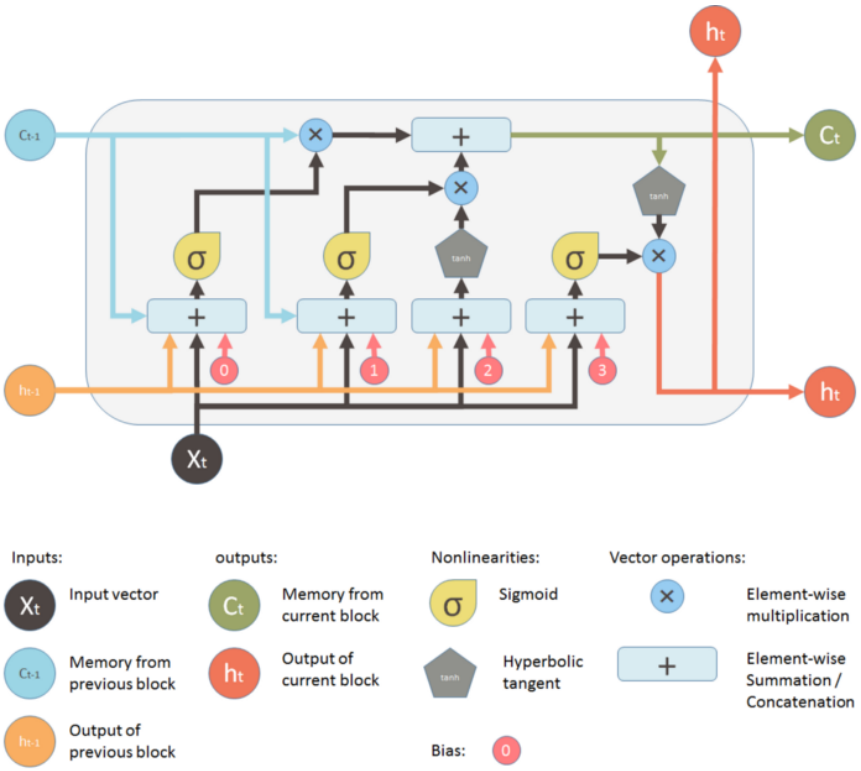


Figure 2: Visual representation of a single LSTM cell. Compare the unfolded contents of the above figure with 1. In the vanilla RNN the hidden layer is updated according to the equations shown in 1. The LSTM replaces this simple constant update at every time step with a logic gate activation method. Note the addition of a persistent cell memory C_t which is selectively updated according to the contents of the forget gate f_t. The selective updating of the cell memory allows the LSTM to retain significant information from the sequence input for longer than a traditional RNN. An LSTM network functions similarly to a RNN network as shown in 1 but with the central cell replaced with a LSTM cell as shown here. Image source: Shi

from the image when entering the LSTM, it may interact with other information free of spatial constraints at each time step. Shi et al. [2015] proposes a further advancement of the LSTM, and implements a fully convolutional LSTM (ConvLSTM). Where the state to state transitions alongside the input to state transitions are all convolutional. The governing equations from 2 are replaced with their convolutional alternatives to give:

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + w_{co} C n_t + b_o) \\
H_t &= o_t \tanh n(C_t)
\end{aligned}
\tag{3}
$$

Here C_t, H_t and X_t are the tensor equivalents of c_t, h_t and x_t in 2, and $*$ denotes a convolution operation. The state to state convolutions in the ConvLSTM act to preserve the spatial structure of the data, in essence limiting the rate at which information can traverse the system at each timestep. A larger convolutional kernel captures higher speed spatial effects. A 1x1 kernel in effect acts as a FC-LSTM. The ConvLSTM has been shown to outperform FC-LSTM in spatio-temporal tasks, as demonstrated in Hong et al. [2017] Gehring et al. [2017]. The structure of the ConvLSTM is appealing to those seeking to predict conflict, due to its capability to combine input from a deep array of predictor variable layers whilst respecting the spatial relationships between the neighbor sites.

3.3 Encoder Decoder

Encoder – Decoder architectures have produced impressive results across a number of sequence prediction activities, especially in natural language processing and translation Neubig [2017]. Encoder – decoder models negate one of the key difficulties in sequence prediction: heterogeneity in input and output lengths. When considering a single reccurent model (whether a vanilla RNN, LSTM, GRU or other variant is immaterial), simply taking the hidden outputs as predictions as in figure 1, produces sequence outputs that must be of equal length to the input sequence.

Encoder decoder architectures were introduced in Sutskever et al. [2014] to counteract these difficulties. Encoder decoder architectures use two deep RNNs. The encoder RNN is fed the input sequence, and the final hidden state h_t of the encoder is used as a fixed size representation of the temporal properties of the input sequence. The hidden state is then copied into the decoder RNN as the initial hidden state and a starting input vector is passed to the decoder. This may be viewed as 'unrolling' the hidden state Park et al. [2018]. Due to the fixed size of the hidden state, variable prediction lengths can be extracted depending on the properties of the decoder. Conditional encoder-decoder models, in which the last output of the encoder is used as the initial input to the decoder in place of a zero centered gaussian input have been shown to produce lower prediction errors than unconditional models Srivastava et al. [2015]. It should be noted that Sutskever et al. [2014] also introduces the concept of reversing the input sequence to LSTM encoders, a very simple idea that has been shown to produce dramatic improvements in performance. However in our experiments we have not seen this effect. We postulate that this is because the increase in performance in LSTMs is mainly seen on sequence translation tasks where there is a far stronger dependence between individual sequence inputs and outputs compared to image prediction. The compressed hidden state representation may also be used to cluster temporal events, an approach demonstrated in Baytas et al. [2017] which also introduces the T-LSTM.

3.4 State of the Art

The Views Project Hegre et al. [2019a] Hegre et al. [2019b] provides a clear example of the state of the art in conflict prediction. The Views Project relies on the PRIO GRID data structure, and is currently the leading publicly available real-time conflict forecasting model. Comparable approaches have previously been developed by the US Dept of Defense and the CIA, but these approaches are not readily accessible. Views combines an ensemble of models based on thematic aggregations of predictors (Baseline, Conflict History, Natural geography, social geography, country level variables and protest) to predict for 36 future months on two scales: on a country by country bases and PRIO grid cell level. They predict the presence of violence (binary) comparing with the UCDP GED database (version 18.1) using dynamic simulation and one step ahead variations of logit and random forest models. see Muchlinski et al. [2016] for a comparison of random forest and logit.
Views has produced several results of note that are worth considering. Primarily they demonstrate that dissagregated gird data outperforms country level data in terms of prediction accuracy (84% vs 99.1% for a full ensemble run) due to the advantages that sub-national dissagregation has when modeling conflict spill over. They also compare the inclusion of a wide number of variables and find the greatest predictive benefit from the inclusion of natural and social geographic features.

Their work also highlights a number of pitfalls with testing and validation which should be considered when applying machine learning techniques to conflict. They isolate the months January 2015 - December 2017 for use as a test set, however they also note the static nature of conflict in Africa in this time frame, implying that their test set may be comparatively "easy", and their accuracy inflated. Even if slightly inflated the accuracies and ROC curves presented demonstrate an incredibly impressive quality of prediction of conflict. Their model both predicts accurately and represents conflict contagion well. We should also note that their use of standardised evaluations of their prediction quality (AUROC, AUPR, Brier score and Accuracy) and pipeline approach allow for easy comparison and bench marking of future conflict prediction efforts. The Views team continue to publish predictions every month which can be found at: https://www.pcr.uu.se/research/views/current-forecasts/

3.5 Datasets

The model uses event and predictor data from the PRIO grid and UCDP-GED data sets Tollefsen et al. [2012a] Sundberg and Melander [2013] . The PRIO (Peace Research Institute of Oslo) data set is widely used in conflict research to provide predictor information at a sub national level. The PRIO grid consists of a vector cell grid of dimensions (0.5 x 0.5) degrees of latitude and longitude covering the globe (approximately $55km \times 55km$ at the equator). Each cell contains variables sorted into conflict relevant groups (socioeconomic, resources and climate.) The UCDP data set (Uppsala Conflict Data Program) records the deaths due to a conflict on a monthly basis. UCDP data are stored as events; an event is a defined as a battle in which violence occurs. For a given political situation to be classed as a conflict, it must meet a threshold of 25 battle related fatalities per year. For a conflict to be recorded all groups involved in the conflict must have a clearly defined identity and a stated goal Croicu and Sundberg [2016]. A version of the UCDP base has been adapted to fit the PRIO grid structure, and covers the years 1989-2018. The UCDP encodes the number of deaths in each grid cell at each month. The norm in conflict prediction is to reduce this to a binary presence variable, where efforts are focused on the prediction of presence of violence, rather than its magnitude. Visualisations of PRIO and UCDP data may be seen in fig 3.

4 Software Development Life Cycle

4.1 Software Requirement Specification

Following the discussion of the project objectives in section 2 a software requirements specification was produced to outline the functionality required to test our hypotheses. As the project followed an Agile methodology, the specifics were left open

ended to allow for iterative improvement.

1. The software must contain a ConvLSTM module implemented using Pytorch. This requirement contains three sub requirements:
   (a) The implementation of the module.
   (b) The implementation of the training and validation functions required to train the model.
   (c) Validation of the produced code against the MovingMNIST data set Srivastava et al. [2015], to ensure our implementation works as required, before exposure to the produced dataset.
   (d) Production of unit tests to verify integrity and continued functionality.
2. The software must contain a pipeline to transform the PRIO and UCDP datasets into sub sampled image formats ready for input into the ConvLSTM model. The output will be a 5th dimensional image tensor of dimensions: (b, s, l, SS_w, SS_h) where b is the batch size, s is the number of steps in the training sequence (i.e the number of preceding months our model will use to predict the next moths conflict), l is the number of different predictor variables we supply to our model, and SS_w and SS_h are the dimensions of the sub sampled input image.
3. The software must allow easy construction of encoder decoder model architectures using the ConvLSTM module implemented in step (1). This should facilitate performance of the model architectures on the dataset constructed in step (2) to be compared to find the optimum configuration.
4. The software must allow data augmentation to improve the robustness of the final model. Special care will be taken with augmentation to respect the norms of social data.
5. The software must contain functionality to compare our predicted results with other predictions in the field, in order to asses the validity of the use of ConvLSTMs to produce conflict, the stated goal of the project.

4.2 Design Rationale

Two rationales were produced to outline the approach to implementing the SRS. The first deals with high level design rationales that broadly describe the approach to the project, whilst the second deals with the rationale behind the implementation itself.

4.2.1 Design Axioms

1. The code must respect and conform to Pytorch practice as closely as possible. This is because:
   (a) We aim to produce a ConvLSTM implementation using Pytorch, an architecture that is not currently provided by Pytorch. While a ConvLSTM implementation is not provided by Pytorch, our implementation may be used in conjunction with other Pytorch modules and should thus follow the same conventions in order to promote ease of use.
   (b) this
2. The code must be as readable and as straightforward as possible. This is because:
   (a) The scope of this project is cross disciplinary, and seeks to bring a new technique into the field of conflict prediction. The code should be easy to read to facilitate analysis of the suitability of the technique, rather than the manner of the implementation.
   (b) Machine learning code is difficult to debug and anticipate. Poor initial performance is expected as one iterates to improve a minimum viable product. The code should be written to facilitate easy fixing.
   (c) The code will be produced in an iterative (Agile) process: straightforward code is straightforward to modify.
3. The code must be structured in a flexible manner to allow for use in future projects. This is because:
   (a) If the proposed method is proved effective, it is likely to be explored in further projects in the conflict prediction community. If this is the case then the code produced in this project may be re used in further works and as a result be used in non anticipated usecases. The code should structured in such a manner to facilitate this.
4. The code production process must prioritise results. This is because:
   (a) The project is exploratory in nature and aimed at evaluating the usecase for ConvLSTMs in the field of conflict prediction.

4.2.2 Approach rationale

Prior to discussing the implementation and results of the project, we present a short summary of the logical steps behind our design choices.

1. The spatial arrangement of economic, social and conflict structures has been shown to have a significant effect on conflict.
2. State of the art conflict prediction machine learning efforts have utilised first order spatial lags to engineer spacial features. Temporal data is predicted using time lagged random forests and logistic regression.
3. Although features are engineered to encode some spatial relationships, the methods used do not respect the spatial encoding of information at either the input to model or the state to state transitions
4. We conjecture that a method that exploits the spatial structuring of conflict data, and restricts the flow of information on a spatial basis in the model, will better model the spatial dynamics of conflict. We also conjecture that using a better performing temporal method will also benefit our prediction.
5. Therefore we require a method that produces significant result in spatio-temporal sequence tasks. We will use the ConvLSTM as:
   (a) ConvLSTM has produced best in class results in image sequence related tasks, including weather prediction Shi et al. [2015]. Video description Srivastava et al. [2015], car crash prediction, and image segmentation.
   (b) ConvLSTM respects the spatial structuring of information at both the input to state and state to state levels. This is a key difference to composite alternatives such as a CNN feature extractor feeding FC-LSTMs Donahue et al. [2017]
   (c) ConvLSTMs are more easily customised than alternatives such as ConvGRUs due to their larger number of input gates, which is important due to the unusual features of conflict data.

4.3 DevOps

A mixture of DevOps tools were used during the project. The project was carried out working as part of the Conflict Analysis Lab (COALA) at UCL. Git interfacing with Github was used for version control and to ensure connection across the two deployment platforms: Google remote machines, and the Imperial College HPC. Travis-CI was used for testing and continuous integration, through the use of unit and integration tests run using Pytest. Google remote storage was used to store datasets and for easy retrieval. All datasets were stored in a HDF5 format to allow storing of meta-data in file, and to allow lazy loading of data samples. Github was used for Issue tracking, and Trello for task management. Communication was carried out via slack and email. Anaconda was used for environment control.

4.4 Design Methodology

The project implementation followed an adaption of the Agile methodology Beck et al. [2001]. As the code was developed standalone, with solely the author writing and using it to produce results, the author simultaneously played the role of both the customer, developer and tester in the feedback process. The role of the manager was performed by the project supervisors. The Agile methodology comprises of short iterations of work during which new features are implemented, then appraised and tested. Due to the exploratory nature of the work, with the goal of testing the suitably of the proposed method to model conflicts, it was vital to have short repeated periods of work following which the performance of the model and functionality implemented so far could be assessed, and the subsequent direction of research altered to take into account conclusions drawn over the previous sprint cycle. As a result, the Agile methodology was found to be the logical choice. The periodisation of the production schedule in Agile was also used to allow for regular input and steering form the project supervisors. While this project seeks to bring a new approach to the field of conflict forecasting, it relies on political, social scientific and economic data and frameworks of assessment, which are often highly nuanced and are benefited by the interpretation of an expert in the field. The use of the Agile methodology allowed for regular retrospectives into the progress that had been made over the previous sprint period, and regular input from the project supervisor to ensure that the political data was interpreted in correctly. Given the relatively short time frame of the project, the project fell into week long iteration cycles between COALA group meetings.

4.5 Code Structure

The project in its current form has been designed as a standalone piece of code. The code is designed to facilitate the easy use of convolutional LSTM architectures for the spatio-temporal prediction of conflicts, and is structured to provide ease of use when writing scripts for use in a research project. The structure of the produced code is in part influenced by environment

in which it was developed and used: primarily in ipython notebooks designed for use on the Imperial College London HPC Cluster. The code produced is structured into two groups, each centered around one of our functional requirements. The first is focused on the production of conflict datasets. The second group implements the classes required to produce ConvLSTM encoder decoder models, and to perform their training and analysis. Each are implemented in separate modules.

4.5.1 Data Production

Methods of converting the PRIO and UCDP data into image sequences were produced in order to provide suitable image sequences for use with a ConvLSTM architecture. The ConvLSTM format uses convolutions to extract structural information from image sequences. We propose to construct a sequence of 'images' corresponding to the value of conflict predictors at each month in Africa, with PRIO grid cells acting as a proxy for image pixels. In place of RGB channels we assign each conflict predictor an image channel, so that each channel is a map of the selected conflict variable across Africa at a given month. The image sequences are then subsampled from the continent wide image sequence by extracting sub image sequence from the larger section. This results in a single image sequence tensor, of dimensions (sequence length, number of channels, image height, image width). This is shown in fig 6. Due to the number of functions involved in the dataset construction, the specifics are better left to the code documentation and examples. Here we simply outline our construction methodology. The implemented functions are used to transform the selected PRIO and UCDP predictors for every month into a sequence of global image maps, with each predictor forming a channel in each map. Random selection functions are then used to identify suitable conflict clusters according to specified criteria and extract the conflict and preceding 10 months of prediction data into an image sequence. This workflow is shown in fig 5. The process is also outlined in pseudo code in algorithms 1 and 2.

4.5.2 ConvLSTM Code

The ConvLSTM encoder decoder model is implemented in three classes in a hierarchical structure. The first two classes, *LSTMunit* and *LSTMmain* are structured as their vanilla Pytorch non convolutional equivalents, following point 1. of the design rationale.
LSTMunit Stores the convolutional operation weights for each layer of the ConvLSTM. The forward method of the ConvLSTM takes an input image, hidden state and cell memory and outputs the subsequent hidden state and cell memory following the theory outlined in equation 3.

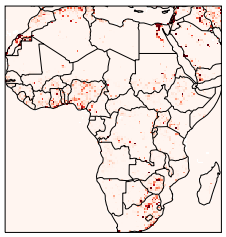
LSTMmain Assembles a number of LSTMunits, with each dedicated to a layer in a deep LSTM. The forward method takes an input image sequence and iterates over the LSTMunits for each layer and timestep. The ConvLSTM allows us to specify which (if any) layers we would like to copy the hidden states into and out from in order to allow for easy construction of encoder - decoder models. A brief pseudo code for the LSTMmain's forward method is shown in algorithm 3

LSTMencdec does not have an equivalent in vanilla Pytorch. The class takes an array detailing the structure of the Encoder - Decoder model. It uses the LSTMmain class to construct two ConvLSTM classes, one encoder and decoder. The encoder, decoder model iterates over two LSTMmain modules, copying the states between the encoder and decoder. The LSTMencdec is a conditional encoder decoder, and is shown in fig 6

A suite of training functions and data loaders we implemented to utilise HDF5 lazy loading to ensure a small load on RAM when drawing from the large datasets. A training wrapper function was implemented to facilitate easy training, and to produce csv log files. Plotting and analysis functions were also implemented and are detailed in the documentation.

4.5.3 Parallelism

ConvLSTM architectures are resource hungry. Due to the high number of channels required for feature extraction Shi et al. [2015] Srivastava et al. [2015], and the temporally deep nature of the input sequences, there is a high resource cost per sample trained. On a single 16GB GPU one epoch took 1.5 hours to complete, with a maximum VRAM usage of 15.5GB when back propagating a 4 layer encoder decoder with hidden channels sizes $(12, 24, 24, 12)$ and a mini batch size of 50 samples. Due to the large number of epochs required to train temporally deep regression problems it became apparent that the model required parallelisation. This was straightforward due to Pytorch's inbuilt parallelisation structure, which follows a distributed architecture, modelled on the MPI protocol Paszke et al. [2017]. Each mini batch is subdivided and scattered from the lead GPU to the subordinate GPUs, where once computed and back propagated the calculated gradients are gathered and summed on the main GPU before the training algorithm performs gradient descent, allowing larger mini batches to be computed per iteration. This significantly reduced training time in the cases presented here, where the computational requirements of the model restrict us to very small mini batch sizes on a single GPU. We note that the Pytorch parallelisation does not speed up the actual calculations of the gradient per GPU, and that too greater a mini batch size will result in poor convergence, due to averaging over local minima in the loss function.



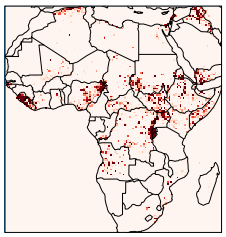


Figure 3: Visualisation of UCDP and PRIO grid predictor data. The left figure visualises the distribution of urban development for each PRIO grid cell in Africa in the year 2000. The right image visualises the frequency of UCDP event occurrence across the entire dataset 1989 - 2014. Both represent image constructions which would each contribute to a channel of image sequence formation, as outlined in section 4.4

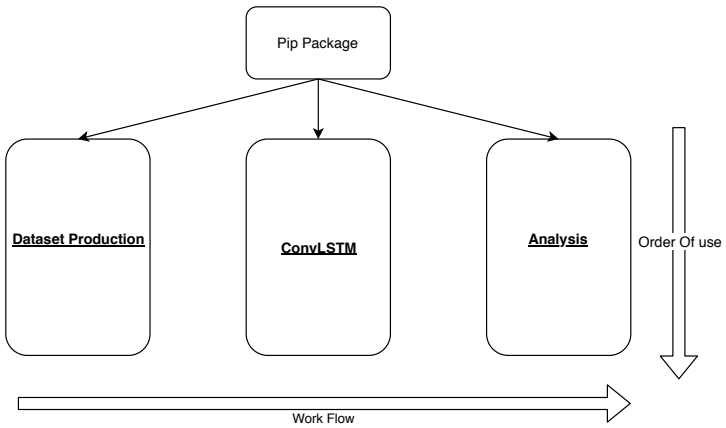


Figure 4: Diagram Showing Structure of produced python package. The package consists of 3 modules, each governing a set functionality.

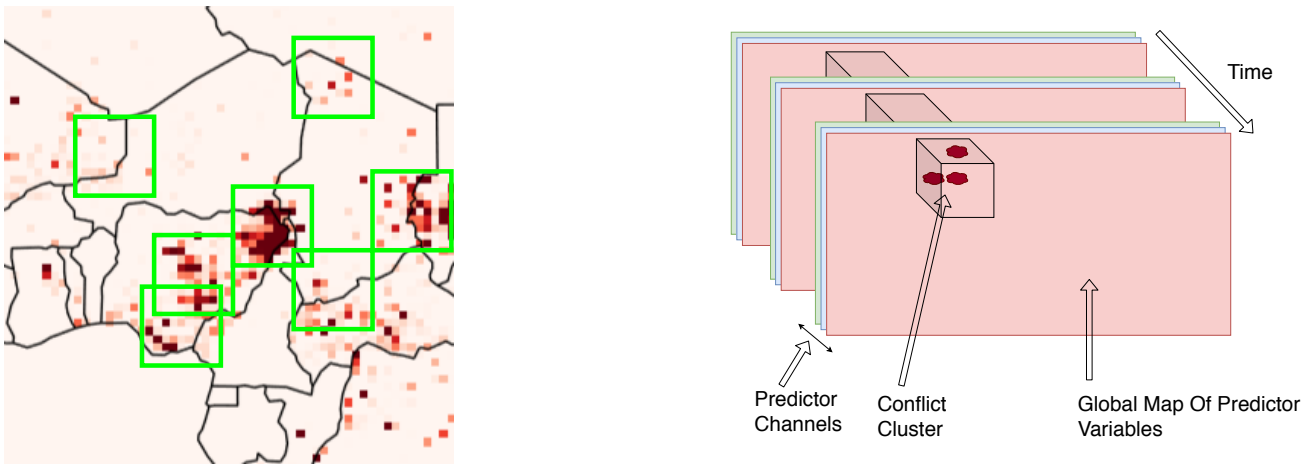


Figure 5: Demonstration of the Image sequence construction process. A sequence of global maps of prio and ucdp predictor variables are constructed and arranged in sequential order. Conflict clusters at each month are identified, with examples of selected clusters shown in the left figure. The predictors in the area surrounding the clusters are extracted for the 10 preceding months, in effect cutting out a slice of the conflict predictor array, as demonstrated in the right figure, to construct a sequence of images encoding the history of the conflict. In the image sequence each chosen predictor is represented by an image channel.

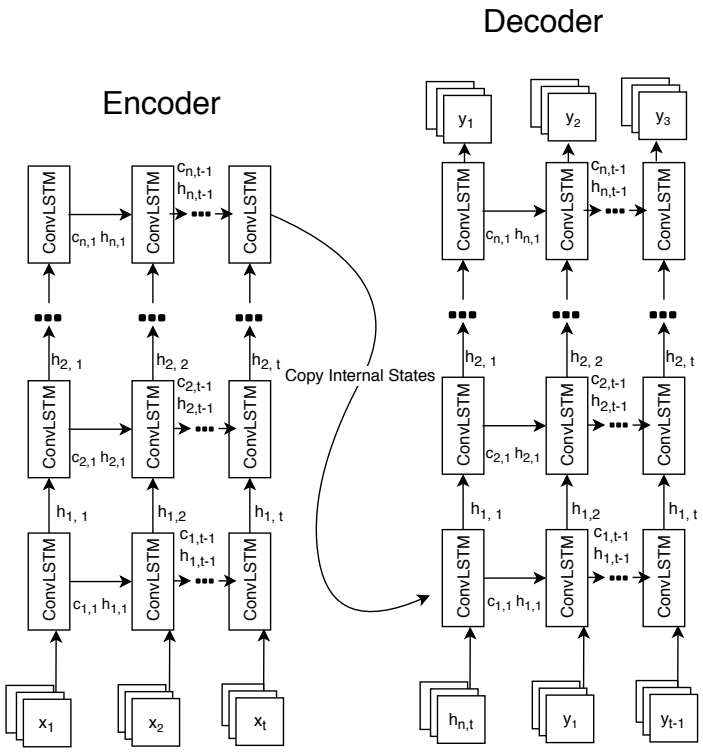


Figure 6: Diagram of the unrolled ConvLSTM encoder decoder model. The encoder decoder is composed of two spatially deep ConvLSTMs, which may be specified as any size. The encoder and decoder are shown as 'unrolled' in the x axis to show the temporal progression of the model as each sample of the sequence input is passed in.

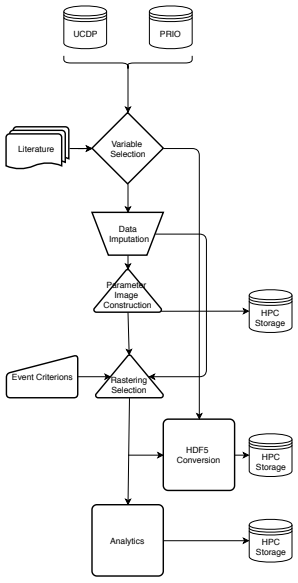


Figure 7: Flow Chart of dataset production process. Data is downloaded from the PRIO and UCDP datasources. Predictors may then be specified from both datasets, before being reshaped into a global parameter image maps. The global image construction and raster selection is outlined in algorithms 1 and 2. Analytics are extracted from the datasets and saved to allow normalisation when the datasets are loaded. The datasets are saved as hdf5 files to allow lazy loading.

Data: UCDP GED data, PRIO grid data 1989 - 2014
Result: Produce 4 dimensional tensor world map of dimensions (months, predictor channels, 360, 720)

```
initialization;
tempmap ← array, dimensions (num months, num channels, 360, 720);
chanlist ← list of variable identifiers for UCDP and PRIO dataframes ;
foreach Event in UCDP GED data do
    foreach channel in chanlist do
        tempmap[event month, channel, PRIO grid x coord, PRIO grid y coord] ← predictor variable for each PRIO
         grid cell ;
    end
end
```

Algorithm 1:

Data: Temp map
Result: Produce data set of subsampled conflict image sequences

```
initialization;
draws ← number of times to randomly sample events ;
chunksize ← dimensions of square cutout shape ;
tempmap ← temporary image map produced in algorithm 1 ;
threshold ← Minimum average number of events in the sequence ;
foreach month in tempmap do
    Kmeans clustering and outlier detection of UCDP events in layer ;
    foreach UCDP event in month layer do
        if event is not an outlier then
            Randomly place cutout raster of dimensions chunksize over event (so that events position is random in
             raster) ;
            Extract raster from 10 previous months. ;
            if Average events in previous sequence > Threshold then
                Dataset ← Preceding sequence, current month cutout
            end
        end
    end
end
```

Algorithm 2:

```
   Data: Input Image Sequence MiniBatch
   Result: Produce Image Prediction
 1 ConvLSTMmain;
 2 Input Sequence ← input image sequence minibatch ;
 3 Layers ← list of ConvLSTMunit in the spatially deep ConvLSTMmain. ;
 4 Copy in layers ← list of hidden states to copy in from an encoder ;
 5 Output ← empty array for outputs ;
 6 h ← list of initial hidden states ;
 7 foreach ConvLSTMunit in Layer do
 8     if A hidden states exists to copy in then
 9         h ← Copy in hidden state ;
10         c ← zero tensor hidden state ;
11     else
12         h ← zero tensor hidden state ;
13         c ← zero tensor hidden state ;
14     end
15 end
16 foreach Image in Input sequence do
17     foreach ConvLSTMunit in layers do
18         h, c ← ConvLSTMunit(Image, h, c) ;
19     end
20     Output ← h ;
21 end
22 return Output
```

Algorithm 3:

4.6 Verification and Validation

Static verification of the code was carried out by a number of self moderated code reviews, to ensure the code conformed well to the design and implementation rationales. As a result of the code reviews a the ConvLSTMmain class' method for copying in a hidden state was altered a number of times, to find a compromise between ease of use and comprehension (point 2) and embodying a structure in line with the vanilla pytorch implementation 1. Dynamic code validation was carried out using Pytests deployed using TRAVIS-CI. This is further discussed in section **??**. As with any piece of research focused software, the main validation tests are the results found when deployed in a real usecase. We therefore present the performance discussed in 6.2 as validation of our implementation.

4.7 Implementation Strategy

Given the choice to explore the use of ConvLSTMs, a brief implementation strategy was developed to allow as straight forward development process as possible. We present a number of key decisions made concerning the implementation, which follow the ethos laid out in sections 4.2.1 and 4.2.2 and aim to fulfill the SRS (section 4.1).

When visually represented conflict data is more sporadic, and follows fewer "well behaved" physical laws than previous ConvLSTM usecases. As a result it was unclear if the produced models would perform well in the conflict usecase. As a result we chose to verify the ConvLSTM on a well explored dummy dataset, to ensure SRS step 1 was completed and verified and a minimum viable product was produced, before moving on to implementation of requirement 2. We also chose to explore the model hyperparameters using the dummy dataset to establish a well performing baseline that could then be a starting point for the hyper parameter search once a conflict dataset was constructed.

Comparison of model structures (requirement 3) was carried out prior to experimentation with dataset design, to ensure the datasets could be compared on a robust, well performing model. In addition we chose to compare the our model to the state of the art (utilising requirement 5) during model experimentation to provide context to our results. To facilitate this requirements $3, 4$ and 5 were implemented at the same time

5 Code Metadata

The code was written as a standalone piece of research software. The code is written in Python, primarily using the Pytorch library. Below is a list of dependencies required, with usecases.

Library	Useage	Version
numpy	numerical manipulation	$\geq$ 1.16.4
torch	model production	$\geq$ 1.1.0
Sci-kit Learn	Feature selection, Metric calculation	$\geq$ 0.21.2
seaborn	Visualisation	$\geq$ 0.9.0
cartopy	Visualitsation	$\geq$ 0.17.0
torchvision	Visualisation of model and debugging	$\geq$ 0.2.2
matplotlib	Visualisation	$\geq$ 2.2.3
pandas	Data processing	$\geq$ 0.25.0
h5py	Data storage	$\geq$ 2.9.0

Table to be filled once requirements finalized.

5.1 Platform

Development was initially carried out using Google Collab interactive notebooks, as procedural validation and testing was far more efficient in an interactive setting, and the GPUs allocated to interactive Collab notebooks on the Google service are considerably more powerful than those allocated to interactive sessions on the Imperial HPC. Following verification the code was transferred to .py scripts and deployed on the HPC, allowing the code to be run in parallel, with multiple jobs running concurrently (both features not available on the Google Collab service.)

5.2 Hardware Requirements

The produced code was collated into a python package, and as a result is operating system agnostic, provided a python environment is accessible. We note that due to both the time and resources required to run we encourage users to run the model remotely on a cluster. The experiments detailed in sections 6 and 7 were carried out on either a single 16GB p100 GPU or 4 4GB p1000 GPUs. We note that the model benefited greatly from parallelisation. Pytorch's GPU acceleration infrastructure is based on the Nvidia CUDA architecture. As a result the model must be run on Nvidia hardware to utilise GPU acceleration. Due to the large number of convolution operations, running without GPU acceleration is unfeasible. TPU acceleration is not currently supported. The code may be found alongside the documentation via Github at: https://github.com/msc-acse/acse-9-independent-research-project-Garethlomax

5.2.1 Data Dependancies

The datasets constructed for use in this project are derived from the PRIO Grid and UCDP GED projects, both of which are updated frequently. For reference, see their respective codebooks Tollefsen et al. [2012a] Tollefsen et al. [2012b] Tollefsen et al. [2012c] Croicu and Sundberg [2016] Conflict and Program [2019]. The VIEWS project may be found at: https://www.pcr.uu.se/research/views/current-forecasts/ The PRIO data project may be found at: https://grid.prio.org/

5.3 Tests

Full system testing is difficult for a machine learning model, as the outcome is not predefined, and is highly unlikely to be a perfect solution to the problem. Metrics such as average mini batch loss or f1 scores can be used to asses model functioning, however often they may not reflect good model performance if the model has converged to a "trivial solution" i.e a solution that optimises the loss function, but not in satisfactory manner to human interpretation. In addition full training, or even model evaluation may not be feasible on test servers due to dependency or hardware requirements. This is true in our case, due to Travis-CI currently not supporting CUDA environments.

As a result the model was first system tested on "human interpretable" datasets from which the performance of the model could clearly be assessed by eye.These are datasets in which the sequence progression can be easily spotted and anticipated by humans, and as such are easier to visually debug the system. The MovingMNSIT dataset Srivastava et al. [2015] was used for early full system tests on the development development platform.

Each Convolutional LSTM class is subject to an initialisation unit test ensuring the constructors at each level are allocating the internal modules correctly. Integration testing is performed at each hierarchical class level, to ensure that any bug can be pinpointed to a specific stage in the integration process. Integration testing is performed using the Pytorch autograd functions, to verify that the function is end to end differentiable, allowing the possibility of testing whether the function is trainable without the large computational overhead of actual training.

Unit tests were also produced for the dataset production process. It should be noted that small unit tests are also preferred for this task as opposed to large full system tests, as the produced global datasets are approximately 50GB in size. As a

result fully testing the produced dataset is not feasible given the size of our CI server. Automatic system tests are run by producing, storing,and reloading dummy datasets.

6 Implementation

6.1 Dataset Construction

The training set is comprised of data from two sources. Data from the Peace Research Institute Oslo (PRIO) and The Uppsala Conflict Data Program (UCDP) was used in the formation of the data set. A brief summary of the data sources and encoding can be seen in section 3.5 While the data in the PRIO and UCDP datasets is spatially encoded, the unit of input to the majority of existing models is at the single PRIO grid cell level, after spatial and temporal lags have been computed. However as noted in section 3 ConvLSTMs require image sequence inputs. Therefore it was required to engineer a new conflict image sequence dataset.

Primarily our objective is to leverage the ability of ConvLSTMs to predict spatio-temporal patterns, to better model the large scale spatial dynamics of conflict. As a result we focus our work predicting the dynamics of widely distributed ongoing conflicts, where we expect these dynamics to be strongest. Isolated conflict events are better predicted using models that can leverage a high amount of categorical and dyadic data, such as the information concerning local disputes between neighbouring towns or communities.Muchlinski et al. [2016]

We select appropriate events by first removing isolated spot events. Conflict clusters were identified by collapsing all conflict events occurring in a given year and then performing Kmeans clustering Hartigan and Wong [1979] coupled with outlier detection on the produced event map. Once events belonging to a cluster were identified, a square grid of length 16 pixels ($\approx 880km$ at equator) was placed over the target and the preceding 10 months conflict events captured into a sequence of preceding images. To prevent bias in the ground truth the grid was placed randomly over the conflict event. To ensure the full scale of the conflict surrounding the event grid cell is captured, and to allow data augmentation, the option to draw multiply randomly placed events was implemented. A thresholding function was implemented to ensure a minimum number of conflict events per conflict sequence. The model was trained on a threshold of 25 conflict events over a 10 month period, with each conflict sequence randomly sampled 4 times. We should note that introducing a threshold introduces a Gaussian probability to the event location in the raster selection, due to the clustered nature of conflict events. By introducing a threshold we ensure that a significant sequence of conflict events was captured, allowing our model to focus on the spatial dynamics. Observe fig 12 for a visualisation of the frequency of conflict events (positive class) per training set ground truth pixel.

6.1.1 Test set Composition

We construct a temporally isolated test set, by separating conflict events from 01/01/2012 to 01/01/2014 from the main training set spanning from 01/01/1989 to 01/01/2012. These bounds are non inclusive, so that the final possible date for a conflict event in the training set is 31/12/2011. This goes against common practice in the field of machine learning, where test sets are usually extracted from the overall data set using a randomly shuffled selection Ripley [2014]. Random construction of the test set is advantageous when assessing the performance of classification models, as it ensures the statistical distribution of the test set features matches those of the training set.

Best practice is less clear with spatio-temporal data. Due to our multiple draw sampling (see section 6.1, and the clustered nature of conflict events in both time and space, we observe that there is likely to be significant overlap between our dataset samples. As a result a random selection is hard to execute without incurring overlap between train and test sets, a worse case scenario in test set construction. In order to have randomly separable samples with no overlap, we must pursue a sub par sampling format. In addition to this we also note that the assertion that the test set must follow the same statistical dynamics as the training set is less applicable in our usecase, due to the usecase of our model.

In our intended usecase the model will be provided the most recent data in order to anticipate future conflict events. As a result we should prioritise the model performance when applied to the most recent data, instead of a random selection of events distributed over the whole duration of the training set to select the most effective models. This is especially relevant once we consider that the internal dynamics of conflict are subject to change over the time and develop as time progresses. As a result we choose to focus model performance on the most recent data, an approach explored in Roberts et al. [2017], which demonstrates that temporally blocked test set data can give better assessments of accuracy for spatio-temporal prediction models.

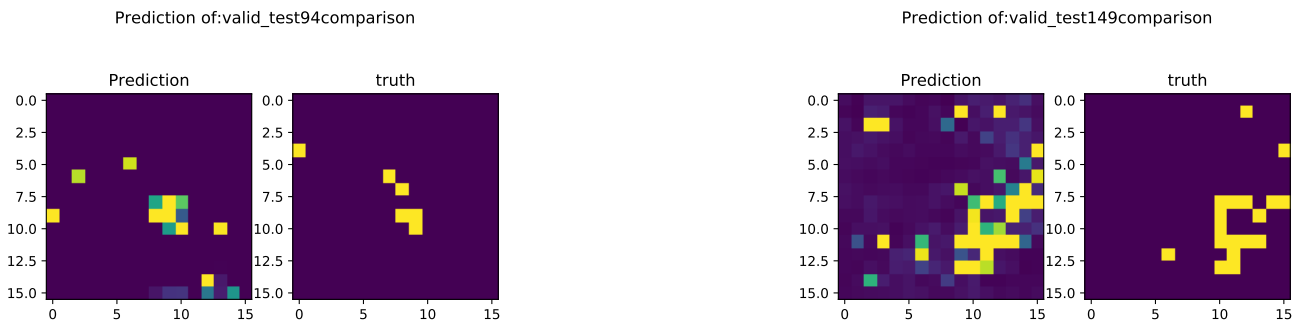


Figure 8: Comparison of the use of BCE and MSE loss functions. In the left figure, BCE is used. The model captures the spatial location of the conflict well but favours recall over precision, leading to blotchy predictions and several surrounding false positives. This may be addressed by scaling the binary class weights. Despite this the prediction is considerably sharper than the comparison on the right, where a MSE loss function is used. Notice both the blurring surrounding the true conflict locations, as discussed in section **??**. Also note the noisy background, which overtakes as the main focus of MSE reduction once initial structural approximations have emerged. This issue is avoided when using BCE loss functions, due to their polarizing effect in separating the prediction classes.

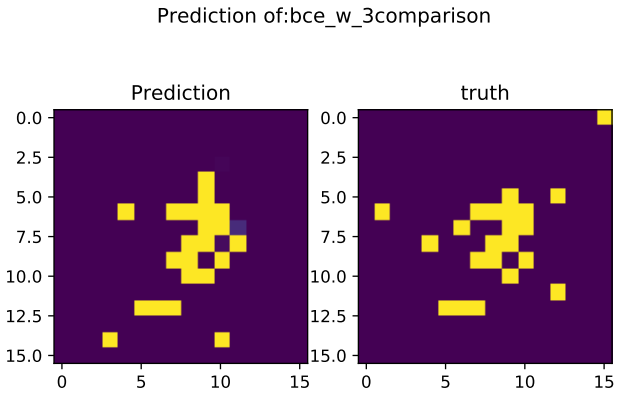


Figure 9: Comparison of sequence prediction using the BCE loss function, with weight scaled by a factor of 3. Note how the ConvLSTM architecture has captured the complex structural arrangement of the conflict in the central mass, and recognises one of two conflict absences surrounded at the center of the conflict cluster.

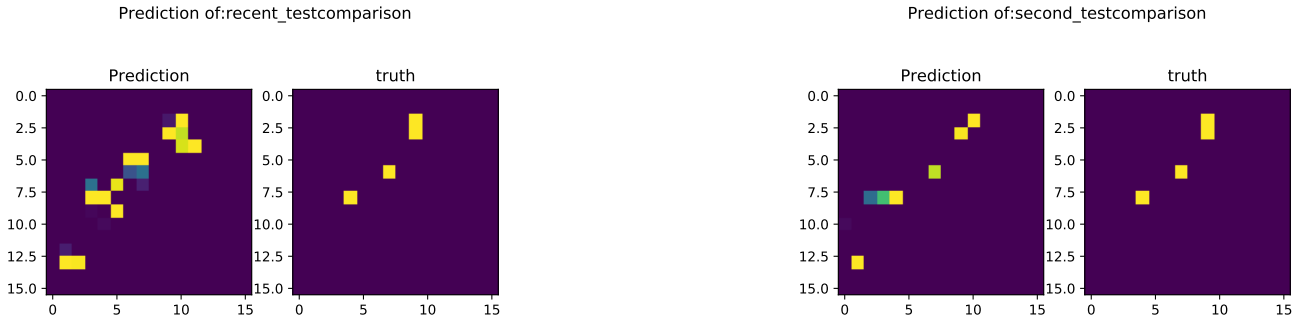


Figure 10: Comparison of a single event prediction using BCE loss function with class weights scaled by an additional factor. Here weights are default as extracted in section **??**.The left image has loss function scaling for each pixel in ratio to the class imbalance. The weights for the right image are reduced by a factor of 3. we observe while the model has anticipated the distribution of the events in both examples, and that the increased recall shown in the left comes at the cost of precision. We can see that the increased precision gives a clearer prediction when the class weights are reduced by scaling.

6.2 Results

6.2.1 Discussion of metrics

A number of performance metrics were used in the assesment of model performance. We define:

$$\text{Accuracy} = \frac{tn + tp}{tn + tp + fp + fn} \tag{4}$$

$$\text{recall} = \frac{tp}{tp + fn} \tag{5}$$

$$\text{precision} = \frac{tp}{tp + fp} \tag{6}$$

$$\text{F1} = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{7}$$

$$\text{Brier} = \frac{1}{N}\Sigma_{i=1}^{N}(p_i - A_i)^2 \tag{8}$$

Where tp, tn, fp, fn are true positive, true negative, false positive and false negative respectively, p_i is the model prediction, A_i is the true function value and N is the number of samples Forman and Scholz [2010]. From equation 4 we can observe that accuracy is a poor metric in class imbalanced cases. In a situation with a large number of negatives, returning a negative in all cases can still give a high accuracy when $tn >> fn, fp, tp$.

Conflict data is inherently class imbalanced. Conflict events are low in frequency and follow an approximate power law distribution in magnitude Friedman [2015].

Fig 12 visualises the number of positive cases for each pixel in our training set ground truth. In 46898 training samples the pixel with peak intensity sees 1780 conflict events, giving us a class imbalance ratio at best of $\approx 3:97$.

Considering the class imbalance we can observe that a standard percentage based accuracy would not provide a suitable metric to many of our classification problems. Trivially predicting no conflict at all would give an accuracy $\approx 97\%$. It is also necessary to consider our design choices in regard to both the precision and recall of our model. Conflict prediction models are increasingly used to direct policy and resources with the aim of mitigating or preventing conflict. As a result there is a high cost associated with false negative results, where an unanticipated conflict could result in a greater loss of life meaning that our model must have a good recall. Prioritising solely the recall of the model however is also non ideal: we must consider that the resources and capability to mitigate a predicted conflict are finite, and often overstretched, and as a result large vague conflict predictions are less useful than precise ones. To this end we adopt the f1 score Forman and Scholz [2010], a metric that balances precision and recall. When calculating f1 score for our multi-labelled comparisons we follow the approach of Forman and Scholz [2010], who investigate best practice of using f1 scores in multi-labelled scenarios.

6.2.2 Overview and State of the art

Our model predicts the next image in the sequence of conflict snapshots passed as input. We visualise these predictions and compare to the ground truth in figures 9 10 and 6.2.These images demonstrate that our model captures the spatial structure of conflict events well, but may predict the exact location less accurately. In particular figure 9 demonstrates the prediction of complex structures well.

We now present our findings and compare our results to the state of the art as discussed in section 3.4. A collection of the model performance metrics shown in table 6.2.2

When used under the right circumstances and usecases we observe impressive model performances when compared to the state of the art. We analyse the model performance on our test set using the F1 score, the Area Under the Reciever Operator Characteristic curve (AUROC), the Area Under the Precision Recall curve (AUPR), and the Brier Score. We find our performance metrics compare well to those presented in the state of the art. The F1 score and AUPR scores of our best model (see table 6.2.2) outperform the VIEWS project who report maximum values of (0.289 and 0.277). The AUROC and Brier scores are worse than the best cases presented in the VIEWS paper (0.9484,0.00623), but still outperform their presented baseline model. We note that the Views model is trained on a far greater selection of conflict predictors than our model, as discussed later.

These assessment metrics are highly encouraging. In particular the F1 scores and AUPR scores, the metrics used which are most suitable for use in class imbalanced conflict prediction Hegre et al. [2019a], denote a significant advance in performance per predictor above the levels reported by the VIEWS team. It should be noted that there are a number of differences between the models which may be relevant in their comparison.

Model	F1 Score	AUROC	AUPR	Brier Score
bce_{w}	0.368125063	0.876813813	0.285600923	0.060787109
bce_{w_2}	0.407514013	0.882485381	0.333934097	0.045832031
bce_{w_3}	0.401870407	0.88845095	0.343948291	0.043720703
bce_{w_4}	N/A	N/A	N/A	N/A
$\text{bce}_{w_2 k_5}$	0.389307649	0.865546798	0.323333498	0.051269531
$\text{bce}_{w_2 k_7}$	N/A	N/A	N/A	N/A
$\text{bce}_{w_3 double}$	N/A	N/A	N/A	N/A
$\text{bce}_{w_3 long}$	N/A	N/A	N/A	N/A
$\text{bce}_{w_3 r}$	0.386544818	0.886919863	0.321480564	0.043134766
$\text{bce}_{w_3 lr=0001}$	0.406102978	0.906116827	0.352464588	0.061884766
$\text{bce}_{w_3 lr=001_A dam}$	0.409390911	0.90669482	0.357358979	0.061171875
$\text{bce}_{w_3 lr=001}$	0.410861686	0.907402586	0.358599216	0.060638672

Table 1: Summary table of model simulation results. In the above table bce_w denotes that the model was run using multilabel adjusted weights (see section 6.2.4). A subscript bce_{w_x} denotes that the weights were reduced by an additional factor x. A subscript k_x denotes that the model was trained with an non default kernel size x (a kernel size of 3 is default.) Subscripts of $lr = x$ denote the learning rate used (default is $lr = 10^{-3}$). The AMSgrad optimiser is used by default. The best performing model $\text{bce}_{3_{lr=001}}$ used a BCE loss function with class imbalance weights reduced by a factor of three, with the AMSgrad Tran and Phong [2019] optimizer using a learning rate of 10^{-3}. The model encoder was a two layer ConvLSTM with hidden channels $(12, 24)$. The decoder had 4 layers with hidden channels: $(24, 12, 6, 2)$. Of the results displayed in the table all results were produced using a BCE multilabel loss function due to the clear performance increase above the other loss functions, as may be seen in the prediction visualisations. N/A denotes the model did not converge, or converged to a trivial prediction.

Firstly, we operate on a reduced dataset of events compared to the VIEWS project. From the full catalogue of conflict events available we sub select conflict clusters of significant size (described in section 3.5), and while the VIEWS project also reduce the size of the their dataset to counteract class imbalance, they do also predict conflicts of smaller scale and duration, which were not included in our training or test data due to their lack of significant spatial dynamics. It is unclear whether the introduction of these events makes their test data harder to predict.

Secondly, we predict events one month in advance, compared to the views project that predicts from 12 to 36 months into the future. Training the produced model to predict longer than a one month period is entirely possible, but was not pursued in this project due to the time limitations on usage of the deployment platform, and the project implementation time frame. We note however that there is little variance in the future conflict dynamics predicted by the VIEWS project, and little change in their performance metrics over the time period of prediction. We note that our one month prediction AUPR and F1 scores match or outperform their one month prediction metrics. The metrics for our prediction also outperform the maximum metrics for any of their prediction months for both AUPR and F1 Score.

Thirdly, our predictions have been made on a far reduced dataset, and still have comparable, or superior performance to the VIEWS predictions. The full range of VIEWS predictors are available in the VIEWS appendix Hegre et al. [2019b]. In their model they utilise 88 predictor variables. Compared to this we utilise 5 predictor variables. This is a noteworthy reduction of the feature space.

6.2.3 Training

It was also observed that the model training is highly susceptible to deep local minima in the model loss function (as may be seen in fig 11). We observe a rapid initial reduction in training and validation loss. Note the difference between losses after a single epoch, with BCE_2 (multilabel binary cross entropy with halved weights converging at a much steeper gradient.) As the weights of the binary cross entropy are reduced by increasing factors, we observe that the model becomes stuck in trivial local minimas. We can observe this in the training curve of BCE_3. The model converges on a trivial local minima (returning a prediction of no conflict for all inputs) in the first epoch. The model does not escape the local minima until the 59th epoch, where it moves to another minima, and produces the best results. Due to limits placed on training time, it is unclear whether BCE and BCE_2 have converged on local or the global minima, and would see a significant loss reduction step comparable to BCE_3 if allowed an extended training period. This will be explored in future work when longer allocations of HPC resources can be secured. The learning rates of the Adam and Amsgrad Tran and Phong [2019] optimisers were then varied . Larger

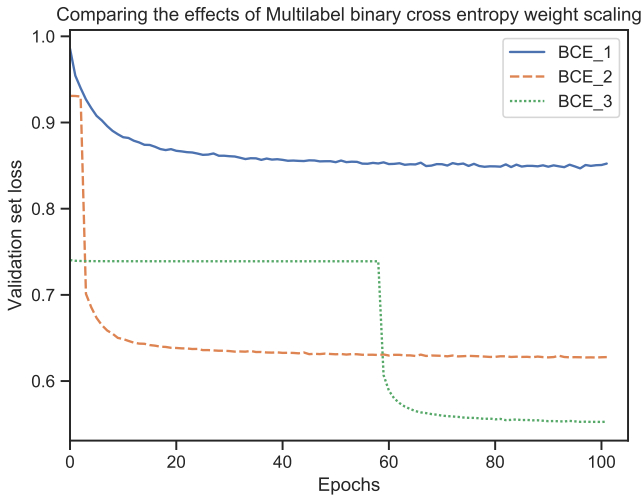


Figure 11: Validation loss during training for BCE loss function with modified weight ratios.

learning rates ($\geq 3 \times 10^{-3}$) were observed to perform poorly, however there was little improvement once the learning rate was reduced past 10^{-3}. In addition the RMSprop optimising algorithm was also used, but did not converge.

6.2.4 Regression vs Labelling

Two possible prediction outputs for the model were explored: a regression of the total number of battle deaths in each PRIO grid cell per month, and a binary indicator of whether a conflict event was predicted to occur in each grid cell. Regression was first implemented using a mean squared error (MSE) loss function, but was found to display poor convergence due to large outlier values caused by the power law distribution of battle deaths in conflict data. Model convergence was significantly improved by instead predicting the binary presence of one or more UCDP events per grid cell. This had two significant advantages: firstly it removed the skew of low frequency high magnitude events dominating the loss function, and secondly it allowed the treatment of the prediction as a multi-label prediction problem, and the introduction of label weights to offset class imbalance by using a binary cross entropy (BCE) multi-label loss function.

MSE was used as the loss function to predict both the binary occurrence of an event per grid cell per month, and the number of battle deaths per grid cell per month. The MSE produced results of varying quality in both the dummy data and real data tasks, as may be seen in figure 6.2. The produced results are 'blurry' with the MSE loss producing blotchy 'over exposed' predictions with large amounts of noise. When the MSE was used in model training an initially rapid drop in training and validation loss was observed before the rate of loss change reduced dramatically. We conjecture that this is as a result of the large amount of uniform blank space in both the dummy and conflict datasets. Once an initial drop in loss has been achieved by producing a rough approximation of the predicted structure high intensity pixels a greater reduction of MSE may be achieved by reducing a large mass of 'no event' pixels predicted to be of an intensity closer to 0. As a result MSE favours rapid convergence on high recall low precision models, which as noted in section **??** are not desirable in conflict prediction.

In addition to this one may note that due to the power law distribution of battle related conflict deaths, the effect of ultra low frequency, high consequence disasters to unnaturally skew the loss function when performing regression on conflict death numbers can be observed, due to the distance between the average model prediction and the ground truth deaths in large scale disasters. For context the events dataset has a mean of 13.688 battle deaths per event and a median of 2.0, whereas the highest consequence single event tragically comprises of 300557 deaths in one event.

The prediction was reframed from a vanilla image regression to binary variable prediction, allowing the use of multi-label classification loss functions. The 16×16 binary image prediction may be viewed as a multi-labeling problem with 256 possible binary categories Zhang and Zhou [2014], with each pixel representing a binary class label. The BCE loss function prioritises separating the two binary classes of each label most effectively and is not dependant on matching the absolute value of the output, merely the degree to which the output may be mapped into distinct distributions for each class label. As a result BCE avoided the over prioritisation of large 'non conflict' spaces that reduced the effectiveness of MSE. We can observe MSE and BCE comparisons in figure 6.2.

To account for class imbalances in the multi-labeling problem, weights corresponding to the inverse of the class frequencies

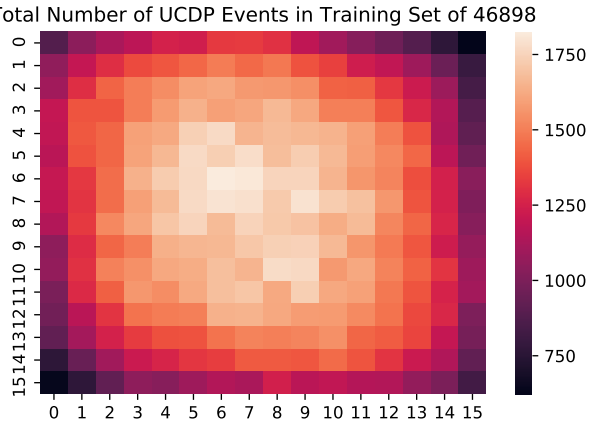


Figure 12: Distribution of the number of conflict events per pixel, compressed over the entire training set. The peak number of events for any pixel is 1780, corresponding to 4% of incidences. Reframing the problem as a multi-lablel prediction exercise allows us to scale the binary cross entropy loss function to reward conflict events and non conflict events in accordance to their ratio of incidence. This ratio can be further multiplied by a scaling factor to skew toward model precision or recall.

displayed in 12 were applied to the loss function labels. This resulted in conflict events predicted correctly (which are ≈ 50 times less frequent) being valued 50 times more than a non conflict event being predicted correctly. This produced far superior results to MSE based training, and so all models included in our comparison with the state of the art rely on BCE. The scaling weights were further divided by factors of $2, 3$ and 4 ($bce_w, bce_{w2}, bce_{w3}, bce_{w4}$ in table 6.2.2. This was found to increase the precision of the model without impacting the recall significantly, leading to increased F1 and AUPR scores($0.368 vs 0.407$, $0.285 vs 0.334$). However reducing by too greater factor (4), produced models that stuck permanently in local minima providing useless results due to over prioritisation of the negative class.

6.2.5 Structure

The structure of the encoder decoder format was varied to find the optimum hyper parameters for model convergence. As in the surrounding literature Shi et al. [2015] Sutskever et al. [2014] Srivastava2015UnsupervisedLSTMs it was observed that deep architectures perform best, but due to the shorter time frame available to this project, architectures that are relatively easy to train were preferred. We use an encoder decoder architecture, with progressively increasing hidden channel size. Increasing the number of hidden channels was found to increase both performance and the number of epochs until convergence. As a result a compromise architecture was used. In further work this would be enlarged to produce results of greater accuracy, but given the project time frame, a smaller model was used to decrease turnaround time. We find the best performing encoder - decoder structure as an encoder with hidden layers with $6, 12, 24$ channels conditionally connected to a decoder with hidden channels of $24, 12, 6, 2$. It was observed that longer decoder structures were resource cheap, when only predicting for short time periods. We observe from table 6.2.2 than reducing the number of hidden channels significantly reduces the performance of a model. In future work models with increased hidden channels will be used.

6.2.6 Kernel Size

Kernel size was found to have variable effect on model performance. Kernel size is significant in the transfer of information in ConvLSTMs. The main advantage of ConvLSTMs over FC-LSTMs are the use of convolutional state to state transitions, which respect the spatial structuring of information in the image. The kernel size of the convolutional transitions may be viewed as a proxy for the velocity at which information may travel at each time step. I.e a larger kernel corresponds to a faster velocity as events further away from a pixel can effect its value at the next time step. A larger kernel size was found to perform well with MSE loss functions, but not BCE. The reason for this is unclear and will be investigated further, however we hypothesise that a large kernel allows more efficient reduction of blank space intensity, which benefits MSE convergence. We find a kernel size of 3 performs best when applied to our dataset. Kernels of size 5 performed well on MSE loss function models, but not BCE. Kernels of size 7 were too large (compared to an image dimension of 16) and did not produce meaningful predictions.

7 Discussion

As discussed in section 6, our implementation clearly performs well under certain circumstances, significantly when operating on a reduced feature dataset. The ConvLSTM model is a powerful tool for predicting conflicts with significant spatial dynamics by leveraging the spatial relation between both the conflict history, and the dynamics of conflict parameters. However the model performs worse in scenarios with small, stop start conflicts. This is theorised to be as a result of the conflicts depending on categorically encoded information, which is expensive to encode in convectional models.

The significant advantage of the ConvLSTM lies in the convolutional operations at the core of our model's functionality. Broadly speaking convolutional neural networks have a high level of information extraction per input feature. This is due to the convolutional filters converging to a set of feature extraction tools that statistically extract the most information out of input images based on the relationships between neighbouring pixels Razavian et al. [2014]. In short convolutional techniques extract information from not only the values of individual predictors, but from the spatial arrangement of the predictors.

Due to this, ConvLSTMs perform well in tasks containing the motion or movement of dynamic structures Donahue et al. [2017]. A majority of the tasks they have been used for are related to the movement and prediction of physical systems Shi et al. [2015]. One of the key questions throughout the project was whether this performance could translate into performance on a social science dataset, where the motion of the conflict is not subject to conservation laws or well behaved physical laws.

We observe however, that our model predicts the movement of conflicts well, and is a powerful tool for assessing the movements and development of medium and large scale conflicts, as illustrated by our high performing F1 and AUPR scores.

The power of the ConvLSTM model is evident when you consider the results presented in section 6. We produce performance metrics comparable to, or better than, the state of the art VIEWS model while using a significantly reduced set of predictors. In the VIEWS paper the authors use up to 88 feature predictors. 4 of these were selected: UCDP best estimate event data encoding UCDP conflict events per prio grid cell, a binary indicator showing whether oil had been produced per grid cell, a binary indicator showing whether drugs were produced in each grid cell, and the mean precipitation per grid cell, and engineer a 5th (binary presence of UCDP event). In other words we achieve comparable 1 month performance utilising $\approx 6\%$ of the available predicting features, by leveraging the hidden information encoded in the spatial structure of the conflict image inputs.

A clear weakness of the model is that it is expensive to leverage categorically encoded information in image format. Many of the available conflict predictors in the PRIO and UCDP datasets are categorically encoded. Furthermore, the categorical variable are often among the most statistically powerful. The models often used in conflict prediction such as logistic regression or boosted random forests work well with categorically encoded information. Categorical information is represented using one hot encoding in machine learning.

In one hot encoding categorical information is represented by a binary vector, with one digit assigned to each category in the piece of information. For example the encoding of a vehicle type variable $x = [\text{orange, apple, banana}]$, a vehicle would be encoded as $x = [1, 0, 0]$ and a lorry $x = [0, 0, 1]$. This representation enforces orthogonality between the categories. It is simple to observe the issue with assigning a singular variable to represent this choice, i.e $\text{orange} = 1, \text{apple} = 2, \text{banana} = 3$ as this would imply the average of an orange and a banana was an apple.

Categorical encoding is a powerful tool, but is rarely used in convolutional neural nets. The corollary of one hot encoding for use with images is to assign a channel to each possible category, with binary encoding of the presence of a category at each pixel. This however leads to a significant parameter growth when even a small number of categories are possible for a parameter. For example, one of the most powerful categorical predictors available to us is the groups or organisations that combatants in a given UCDP event belong to. This allows us to differentiate between different conflicts in the same location. However there are 748 groups present in the data. Producing an image channel encoding of this size is clearly unfeasible. Before categorical information is to be encoded we must reduce the feature space, however with social science data this is often a complex procedure. For many potentially useful variables more work must be carried out to ensure we engineer our predictors properly.

In addition to this the ConvLSTM samples a large variety of data points in a single prediction, due to the image snapshots encompassing a 16×16 grid of PRIO grid cells. This is compared to the state of the art VIEWS project, which uses logistic regression, random forests and model dynamism. These methods account for spatial effects by using first order spatial lags to engineer features, resulting in a data that is reliant on, at largest, a 3×3 PRIO grid cell space. Missing or incomplete data, which is understandably common in continent wide collection efforts has a far greater effect on the data structure of our model than the state of the art due to the sampling structure. As the random forest and logistic regression models predict for single data points at a time (ignoring mini batch size) simple listwise deletion or single valued imputation can be carried out. Our method however requires that each input of size 16×16 (or $\approx 880km \times 880km$) PRIO grid cells to contain no missing parameters over a 11 month duration. Clearly this means our datasets are far more sensitive to missing data than

comparable datasets, and so require an increased level of data imputation to produce datasets of comparable breadth.

7.1 Appraisal of Approach

While the project has produced a number of encouraging results, there is still much to be learned both in terms of the model implementation, and in the manner in which the model was implemented. An Agile methodology was chosen from the onset of the project production. While the Agile method was useful for producing result orientated code, its use resulted in a number of issues towards the end of the project. Agile does not prioritise documentation, or *polished* code. These can usually be improved upon at the end of a project, however in this case a large flaw in the training data was discovered a week before the submission deadline, a period which had been set aside dedicated to improving the software sustainability after the finish of the code implementation and analysis of results. While the flaw was fixed, it was time consuming to locate, and meant that all results generated up to this point in the project had to be discarded and re generated, resulting in insufficient time to implement the neglected software sustainability. A waterfall approach, in which documentation and sustainability is prioritised would have been more robust against such a flaw. While one could argue that less promising results may have been produced using a waterfall methodology, it would mean that the final delivered product was a more well rounded software package. Given the promising results displayed, the author intends to continue work on the project following the formal submission, and will prioritise a rapid increase of software sustainability.

One may also note the cross disciplinary nature of the project. The author is pleased that a self driven and proposed project has produced exciting preliminary results, however it should be noted that the significant time required to become familiar with a new field of research resulted in less progress than the author expected.

7.2 Future Work

The approach of treating conflict data as an image based problem is novel, despite the suitability of the data structures of several large scale data collection efforts. The use of ConvLSTMs and image based machine learning techniques in general have been demonstrated to be a promising area of research in conflict prediction. We aim to explore a number of possibilities further.

The VIEWS project is now focused on producing a platform on which other prediction models may be deployed and combined into their ensemble modelling protocol. Once this platform is in production, we anticipate refactoring the model for use and comparison on their platform.

In addition to this we also aim to explore the effect of experimenting with a greater range of conflict predictors to improve model performance. This was originally planned to be done as part of the project, but given the complexity of imputation required it will be pursued after project completion to allow time for best practice to be followed.

Finally we aim to explore the application of Sohl-Dickstein and Kawaguchi [2019], which derives a loss function scaling that can eliminate all bad local minima. As our model is highly susceptible to local minima we anticipate that this will greatly benefit model training.

7.3 Conclusion

A small Pytorch addon package was produced to allow researchers to easily construct Convolutional LSTM encoder - decoder models, a functionality not currently implemented in vanilla Pytorch. The ConvLSTM implementation is versatile and applicable to any image sequence task. The overall package is designed to allow conflict researchers to be able to easily explore the role of ConvLSTMs in conflict: to this end a suite of functions for constructing image sequence datasets from conflict data was implemented. Plotting and analytic functions were also implemented. The produced package is fully documented with a suite of CI pytests running on Travis-CI. The produced code was used to asses the suitability of ConvLSTMs for use in conflict prediction. Using a simple Convolutional encoder decoder model we produce predictions whose performance metrics match or exceed the current state of the art, when using $\approx 6\%$ of the features as required by previous state of the art predictions. This clearly demonstrates that Convolutional LSTM models have great utility in the field of conflict prediction and should be explored further.

References

Inci M. Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K. Jain, and Jiayu Zhou. Patient Subtyping via Time-Aware LSTM Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, volume Part F1296, pages 65–74, New York, New York, USA, 2017. ACM Press. ISBN 9781450348874. doi: 10.1145/3097983.3097997. URL http://dl.acm.org/citation.cfm?doid=3097983.3097997.

K Beck, M Beedle, A Van Bennekum, A Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for Agile Software Development, 2001.

Anjali Thomas Bohlken and Ernest John Sergenti. Economic growth and ethnic violence: An empirical investigation of Hindu-Muslim riots in India. *Journal of Peace Research*, 47(5 Sep 2010):589–600, 2010. ISSN 00223433. doi: 10.1177/0022343310373032.

Nils-christian Bormann, Luc Girardin, Philipp Hunziker, Manuel Vogt, and Group-level Data. GROW up Research Front-End EPR CODE BOOK. 2018.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. pages 1–9, 2014. URL http://arxiv.org/abs/1412.3555.

Paul Collier and Anke Hoeffler. On incidence of civil war in Africa. *Center for Development Research (ZEF Bonn)*, 1 (December):1–27, 2000. ISSN 1098-6596. doi: 10.1017/CBO9781107415324.004.

Uppsala Conflict and Data Program. UCDP / PRIO Armed Conflict Dataset Codebook. 39(2002), 2019.

Mihai Croicu and Kristine Eck. ViEWS. 694640:1–9, 2018.

Mihai Croicu and Ralph Sundberg. UCDP GED Codebook Version 17.1. *Journal of Peace Research*, 50(4):523–532, 2017. URL http://ucdp.uu.se/downloads/ged/ged171.pdf.

Mihai Catalin Croicu and Ralph Sundberg. UCDP Georeferenced Event Dataset Codebook Quick Start Guide. (October), 2016.

Zhiyong Cui, Ruimin Ke, and Yinhai Wang. Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. pages 22–25, 1 2018. URL http://arxiv.org/abs/1801.02143.

Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2599174.

James D. Fearon and David D. Laitin. Ethnicity, insurgency, and civil war. *American Political Science Review*, 97(1):75–90, 2003. ISSN 15375943. doi: 10.1017/S0003055403000534.

George Forman and Martin Scholz. Apples-to-apples in cross-validation studies. *ACM SIGKDD Explorations Newsletter*, 12 (1):49, 11 2010. ISSN 19310145. doi: 10.1145/1882471.1882479. URL http://portal.acm.org/citation.cfm?doid=1882471.1882479.

Jeffrey A. Friedman. Using Power Laws to Estimate Conflict Size. *Journal of Conflict Resolution*, 59(7):1216–1241, 10 2015. ISSN 0022-0027. doi: 10.1177/0022002714530430. URL http://journals.sagepub.com/doi/10.1177/0022002714530430.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional Sequence to Sequence Learning. 2017. URL http://arxiv.org/abs/1705.03122.

Alex Graves. Supervised Sequence Labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, chapter 3, 4, pages 18–38. Springer-Verlag Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-24797-2{_}2. URL http://link.springer.com/10.1007/978-3-642-24797-2_2.

Alex Graves. Generating Sequences With Recurrent Neural Networks. pages 1–43, 2013. URL http://arxiv.org/abs/1308.0850.

Weisi Guo, Xueke Lu, Guillem Mosquera Donate, and Samuel Johnson. The Spatial Ecology of War and Peace. *Arxiv Preprint: https://arxiv.org/abs/1604.01693*, 2017(June):1–9, 2016. URL http://arxiv.org/abs/1604.01693.

J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. *Applied Statistics*, 1979. ISSN 00359254. doi: 10.2307/2346830.

Hvard Hegre and Nicholas Sambanis. Sensitivity analysis of empirical results on civil war onset. *Journal of Conflict Resolution*, 50(4):508–535, 2006. ISSN 00220027. doi: 10.1177/0022002706289303.

Håvard Hegre, Nils W. Metternich, Håvard Mokleiv Nygård, and Julian Wucherpfennig. Introduction: Forecasting in peace research. *Journal of Peace Research*, 54(2):113–124, 2017. ISSN 14603578. doi: 10.1177/0022343317691330.

Håvard Hegre, Marie Allansson, Matthias Basedau, Michael Colaresi, Mihai Croicu, Hanne Fjelde, Frederick Hoyles, Lisa Hultman, Stina Högbladh, Remco Jansen, Naima Mouhleb, Sayyed Auwn Muhammad, Desirée Nilsson, Håvard Mokleiv Nygård, Gudlaug Olafsdottir, Kristina Petrova, David Randahl, Espen Geelmuyden Rød, Gerald Schneider, Nina von Uexkull, and Jonas Vestby. ViEWS: A political violence early-warning system. *Journal of Peace Research*, 56(2):155–174, 2019a. ISSN 14603578. doi: 10.1177/0022343319823860.

Håvard Hegre, Marie Allansson, Matthias Basedau, Michael Colaresi, Mihai Croicu, Hanne Fjelde, Frederick Hoyles, Lisa Hultman, Stina Högbladh, Remco Jansen, Naima Mouhleb, Sayyed Auwn Muhammad, Desirée Nilsson, Håvard Mokleiv Nygård, Gudlaug Olafsdottir, Kristina Petrova, David Randahl, Espen Geelmuyden Rød, Gerald Schneider, Nina von Uexkull, and Jonas Vestby. ViEWS: A political violence early-warning system. *Journal of Peace Research*, 56(2):155–174, 2019b. ISSN 14603578. doi: 10.1177/0022343319823860.

Sepp Hochreiter. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 2003. ISSN 0218-4885. doi: 10.1142/s0218488598000094.

Seungkyun Hong, Seongchan Kim, Minsu Joh, and Sa-kwang Song. PSIque: Next Sequence Prediction of Satellite Images using a Convolutional Sequence-to-Sequence Network. *Arxiv Preprint arXiv:1711.10644*, (Dlps):1–5, 2017. URL http://arxiv.org/abs/1711.10644.

Nils W. Metternich, Shahryar Minhas, and Michael D. Ward. Firewall? or Wall on Fire? A Unified Framework of Conflict Contagion and the Role of Ethnic Exclusion. *Journal of Conflict Resolution*, 61(6):1151–1173, 7 2017. ISSN 0022-0027. doi: 10.1177/0022002715603452. URL http://journals.sagepub.com/doi/10.1177/0022002715603452.

Anirban Mitra and Debraj Ray. Implications of an Economic Theory of Conflict: Hindu-Muslim Violence in India. *Journal of Political Economy*, 122(4):719–765, 8 2014. ISSN 0022-3808. doi: 10.1086/676316. URL https://www.journals.uchicago.edu/doi/10.1086/676316.

David Muchlinski, David Siroky, Jingrui He, and Matthew Kocher. Comparing Random Forest with Logistic Regression for Predicting Class-Imbalanced Civil War Onset Data. *Political Analysis*, 24(1):87–103, 1 2016. ISSN 1047-1987. doi: 10.1093/pan/mpv024. URL https://www.cambridge.org/core/product/identifier/S1047198700012055/type/journal_article.

Graham Neubig. Neural Machine Translation and Sequence-to-sequence Models: A Tutorial. pages 1–65, 3 2017. URL http://arxiv.org/abs/1703.01619.

Sean P. O'Brien. Crisis early warning and decision support: Contemporary approaches and thoughts on future research. *International Studies Review*, 2010. ISSN 15219488. doi: 10.1111/j.1468-2486.2009.00914.x.

Ola Olsson. Conflict diamonds. *Journal of Development Economics*, 82(2):267–286, 3 2007. ISSN 03043878. doi: 10.1016/j.jdeveco.2005.07.004. URL https://linkinghub.elsevier.com/retrieve/pii/S0304387806000447.

Seong Hyeon Park, Byeongdo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2018-June:1672–1678, 2018. doi: 10.1109/IVS.2018.8500658.

Adam Paszke, Gregory Chanan, Zeming Lin, Sam Gross, Edward Yang, Luca Antiga, and Zachary Devito. Automatic differentiation in PyTorch. *31st Conference on Neural Information Processing Systems*, 2017. ISSN 1098-6596. doi: 10.1017/CBO9781107707221.009.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2014. ISBN 9781479943098. doi: 10.1109/CVPRW.2014.131.

Brian D. Ripley. *Pattern recognition and neural networks*. 2014. ISBN 9780511812651. doi: 10.1017/CBO9780511812651.

David R. Roberts, Volker Bahn, Simone Ciuti, Mark S. Boyce, Jane Elith, Gurutzeta Guillera-Arroita, Severin Hauenstein, José J. Lahoz-Monfort, Boris Schröder, Wilfried Thuiller, David I. Warton, Brendan A. Wintle, Florian Hartig, and Carsten F. Dormann. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. *Ecography*, 40(8):913–929, 8 2017. ISSN 09067590. doi: 10.1111/ecog.02881. URL http://doi.wiley.com/10.1111/ecog.02881.

Pau Rodríguez, Miguel A. Bautista, Jordi Gonzàlez, and Sergio Escalera. Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing*, 2018. ISSN 02628856. doi: 10.1016/j.imavis.2018.04.004.

Heather Sarsons. Rainfall and conflict: A cautionary tale. *Journal of Development Economics*, 115(July):62–72, 7 2015. ISSN 03043878. doi: 10.1016/j.jdeveco.2014.12.007. URL https://linkinghub.elsevier.com/retrieve/pii/S030438781400159X.

Neha Sharma, Vibhor Jain, and Anju Mishra. An Analysis of Convolutional Neural Networks for Image Classification. In *Procedia Computer Science*, 2018. doi: 10.1016/j.procs.2018.05.198.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 802–810, 2015. URL http://arxiv.org/abs/1506.04214.

Yan Shi. Understanding LSTM and its diagrams. URL https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714.

Jascha Sohl-Dickstein and Kenji Kawaguchi. Eliminating all bad Local Minima from Loss Landscapes without even adding an Extra Unit. page 2, 1 2019. URL http://arxiv.org/abs/1901.03909.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised Learning of Video Representations using LSTMs. In *ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning - Vol 37*, pages 843–852, 2015. URL http://arxiv.org/abs/1502.04681.

Ralph Sundberg and Erik Melander. Introducing the UCDP Georeferenced Event Dataset. *Journal of Peace Research*, 50 (4):523–532, 7 2013. ISSN 0022-3433. doi: 10.1177/0022343313484347. URL http://journals.sagepub.com/doi/10.1177/0022343313484347.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *Arxiv Preprint*, pages 1–9, 9 2014. URL http://arxiv.org/abs/1409.3215.

Philip E. Tetlock. *Expert Political Judgment*. Princeton University Press, Princeton, 12 2018. ISBN 9781400888818. doi: 10.1515/9781400888818. URL http://www.degruyter.com/view/books/9781400888818/9781400888818/9781400888818.xml.

Andreas Forø Tollefsen, Håvard Strand, and Halvard Buhaug. PRIO-GRID: A unified spatial data structure. *Journal of Peace Research*, 49(2):363–374, 2012a. ISSN 00223433. doi: 10.1177/0022343311431287.

Andreas Forø Tollefsen, Håvard Strand, and Halvard Buhaug. PRIO-GRID: A unified spatial data structure. *Journal of Peace Research*, 49(2):363–374, 2012b. ISSN 00223433. doi: 10.1177/0022343311431287.

Andreas Forø Tollefsen, Håvard Strand, and Halvard Buhaug. PRIO-GRID: A unified spatial data structure. *Journal of Peace Research*, 49(2):363–374, 2012c. ISSN 00223433. doi: 10.1177/0022343311431287.

Phuong Thi Tran and Le Trieu Phong. On the Convergence Proof of AMSGrad and a New Version. *IEEE Access*, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2916341.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:3156–3164, 2015. ISSN 10636919. doi: 10.1109/CVPR.2015.7298935.

Nils B. Weidmann. Violence "from above" or "from below"? The role of ethnicity in Bosnia's civil war. *Journal of Politics*, 2011. ISSN 00223816. doi: 10.1017/S0022381611000831.

Nils B. Weidmann and Sebastian Schutte. Using night light emissions for the prediction of local wealth. *Journal of Peace Research*, 2017. ISSN 14603578. doi: 10.1177/0022343316630359.

Nils B. Weidmann and Michael D. Ward. Predicting conflict in space and time. *Journal of Conflict Resolution*, 54(6):883–901, 2010. ISSN 15528766. doi: 10.1177/0022002710371669.

Min Ling Zhang and Zhi Hua Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014. ISSN 10414347. doi: 10.1109/TKDE.2013.39.