A Graph Neural Network Approach to Wildfire Cause Prediction

Mark L. Tenzer*
James M. Howerton*
mlt2jf@virginia.edu
jh3df@virginia.edu
University of Virginia School of Data Science
Charlottesville, Virginia

ABSTRACT

As a means to aid those who will be tasked with investigating the wildfires of tomorrow, we propose a new methodology for analyzing wildfire data and predicting wildfire cause. Previous attempts to predict wildfires' causes have largely ignored information from recent fires in the area. Our novel approach applies various graph-neural-network (GNN) techniques involving information about each individual fire, in addition to fires close to it spatially and temporally, to predict the cause of any given wildfire. We utilize 24 years of geo-referenced wildfire data augmented with global population center data and NOAA GSOD weather data.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Supervised learning by classification*; • **Applied computing** → *Earth and atmospheric sciences*; **Investigation techniques**.

KEYWORDS

graph neural networks, node classification, wildfires, public safety

ACM Reference Format:
Mark L. Tenzer and James M. Howerton. 2020. A Graph Neural Network Approach to Wildfire Cause Prediction. In *Proceedings of Graph Mining*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 STATEMENT OF PROBLEM AND LITERATURE REVIEW

The state-of-the-art in wildfire cause determination is exhaustive government investigation. The US government-provided guide, the Guide to Wildland Fire Origin and Cause Determination [9], contains hundreds of pages that provide in painstaking detail the specific indicators of wildfire behavior, on-the-ground evidence at the fire's origin, and witness testimony. Evidence must be carefully collected, documented, and preserved, and may have been scorched—indeed, in the event that the fire was caused by arson, an investigation of a wildfire's cause is fundamentally a criminal case. The hazardous conditions of a wildfire often delay and restrict investigators' access to key sites, and once they do gain access, the investigation is an arduous process. It would be immensely beneficial to gain an early indication of the likely causes of a fire, to be validated later by formal investigation.

Attempts have been made to model wildfire cause and distribution [1, 11]; however, to our knowledge, this is the first attempt to incorporate information from related fires via a graph structure. We propose applying new methodologies including graph neural networks to this domain. Although there are a variety of graph neural network approaches, we aim to utilize convolutional graph neural networks, due to our belief that the local information about nearby fires can provide greater insight than if the fires were considered globally.

Prior research has found that the large majority of US wildfires are caused by humans in the surrounding areas: 84 percent of fires are caused by human activity, tripling the length of the annual fire season, especially outside of sparsely populated areas [4]. To incorporate this useful predictor into our model, we augmented our wildfire dataset to include information about the closest community hubs, their distances, and their populations. We hoped that this data augmentation would provide a greater level of accuracy than previous methods. Prior research has also found that elevation is a key factor in wildfire prediction [4, 11], potentially because lightning can more easily reach high elevations. Additionally, moisture should also be predictive of a wildfire's cause, because empirically, it has been shown that lightning-related fires tend to occur in drier environments than human-activity-related fires [4]. As a result, we augmented our dataset to include the elevation and weather of the wildfire using National Oceanic and Atmospheric Administration (NOAA) weather station data [2].

2 DATASET DESCRIPTION

2.1 Primary Data Acquisition

The dataset for this task was derived from the Forest Service Research Data Archive [1, 10], containing data on approximately 1.88 million wildfires occurring in the United States from 1992 to 2015, sourced from Kaggle. Correspondingly, the graph for this problem contained $|V| \approx 1.88$ million nodes. The edges were not explicitly specified with the provided data; instead, the graph was formed by creating an undirected edge between any pair of wildfires that occur within a distance of D kilometers and time difference of T days.

At each node in the graph, representing one wildfire and connected to all related fires, was a feature vector describing relevant information about the wildfire, including its starting date (time of

*Both authors contributed equally to this research.

Graph Mining, May 2020, Charlottesville, VA, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

day, day of week, month, year), containment date, and duration; its size (in acres); and its location (latitude, longitude, state, county, broad geographic region within the US).

At each node in the graph, there was also a categorical label with 11 mutually-exclusive possibilities: lightning, debris burning, campfire, equipment use, arson, children, railroad, smoking, powerline, structure (e.g., a house or other structure caught fire), or fireworks. Fires coded as "miscellaneous" or "missing/undefined" were considered unlabeled, making this nominally a semi-supervised node classification problem. However, in most semi-supervised contexts, there are more unlabeled observations than labeled ones, whereas here only about 500,000 fires, of almost 2 million, were of unknown cause.

2.2 Data Cleaning & Augmentation

2.2.1 WCD Data. The World Cities Database (WCD) [3] contains features about numerous cities throughout the world, but most relevant for wildfire prediction were their locations and populations. This information was used to calculate the distance to the nearest cities of various sizes, from small towns to major cities, for each wildfire. Four columns were added to the dataset for the distance to the closest city of size $\geq$ 1,000,000, 100,000, 10,000, and 1,000. For each of these four sets of cities, we constructed a k-d tree with the latitude and longitude of the fire and of the city in the WCD. k-d trees allow very efficient nearest-neighbor searches in multidimensional spaces [5].

2.2.2 NOAA GSOD Weather Database. Our wildfire data has been augmented with the NOAA[2] GSOD weather database to include the elevation and weather at each fire's initial location and time. This was accomplished using the latitude, longitude, and day of year of both the fire and the weather station recording. To perform the augmentation, we used k-d trees in a similar fashion as we did with the WCD augmentation and, to make the problem more feasible, we only matched entries that occurred within the same year. Even with this more efficient algorithm, merging the datasets still required hours of computation.

2.2.3 Adjacency Matrix Formulation. Because this dataset was heavily augmented for this project and did not contain a preexisting graph structure or adjacency matrix, it was necessary to decide what attributes to use to connect the nodes of the graph (the wildfires). Wildfires could be connected on the basis of time: if two wildfires occurred simultaneously, they were likely influenced by the same seasonality and weather patterns. They could also be connected on the basis of location: if two wildfires occurred in the same area, they were more likely to have similar causes (e.g., lightning is more common in some areas). However, using either approach in isolation appeared insufficient. Such graphs were highly densely connected, producing adjacency matrices too large to fit in memory, process, or model useful relationships. They also exhibited unusual degree distributions for real-world graphs.

Using only wildfires from 2015 (to limit the size of the adjacency matrix temporarily), and a time interval of $T = 1$ days (the smallest interval that could make sense on this dataset), still resulted in an average degree of $c = 807.9$. Additionally, this degree distribution was approximately Normally distributed. Using a spatial distance of $D = 1^\circ$ for GPS coordinates, the average node degree was a similarly high $c = 749.2$. However, requiring a connected pair of wildfires to meet *both* conditions resulted in an approximately power law distribution and a sparse graph, with average node degree of $c = 21.2$.

To calculate these results efficiently on the full 1992–2015 dataset, we designed our own algorithm to calculate all connections between the nodes of our augmented wildfire dataset in near-linear time. Calculating whether two nodes were adjacent in time and GPS location fundamentally required calculating two distance matrices (one containing time intervals, and the other Euclidean spatial distances), thresholding both, and taking only elements equal to 1 in both (element-wise multiplication). However, with a large node set ($|V| \approx 1.88$ million), naïvely calculating the distance matrices was not computationally feasible, as it required prohibitive computational and memory resources to calculate and store over 3.5 trillion elements.

The solution required the insight that if the matrix was partitioned into many smaller "blocks,"

$$A = \left[\begin{array}{c|c|c} A_{11} & \dots & A_{1N} \\ \hline \vdots & \dots & \vdots \\ \hline A_{N1} & \dots & A_{NN} \end{array}\right]$$

these separate components could be calculated separately. Additionally, since a pair of wildfires must be close in both time and GPS space to be considered connected in this graph, if one block's distance matrix contained no non-zero elements for time, the spatial distance matrix calculation could be skipped altogether. Moreover, since all edges were undirected, it was not necessary to calculate roughly half of the elements: they were identical to the transposed block of the matrix, $A_{ij} = A_{ji}$.

Therefore, the wildfires were sorted by their day of discovery, using a simple mergesort for its stability. The matrix was then partitioned into 100×100 blocks (allowing smaller blocks at the end of a row or column, since $|V|$ was not divisible by 100). Beginning in the top-left block A_{11}, each block's time distance matrix was thresholded at time intervals of $T = 1$ day, considering one wildfire connected to a second one if it occurred on the previous, same, or next day. If it contained any non-zero elements, the spatial distance matrix was then calculated and thresholded at a distance of $D = 1^\circ$ degrees. The element-wise product was then stored in A and the algorithm proceeded to the next block in the given row. Using element-wise products here, since all elements were 0 (False) or 1 (True), guaranteed that both conditions are met.

However, as the algorithm iterated along the row, the time intervals increased monotonically (because the fires were sorted by time); it would be unnecessary, for example, to calculate A_{1N} because this block must contain time intervals of years! Therefore, all non-zero elements must lie close to the sorted A's diagonal. When a block A_{ij} was found that contained all zeros, all remaining calculations on this row were skipped, and the algorithm resumed at the next block on the diagonal, $A_{i+1,i+1}$.

Initial construction of the matrix was performed using a list-of-lists (LIL) sparse matrix; all sparse matrices have much lower memory requirements than ordinary dense matrices, but LIL is particularly well-suited to fast queries of multiple points along

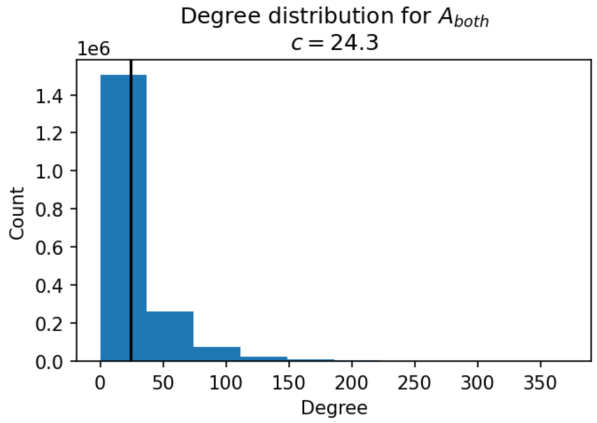

Figure 1: Degree distribution for the full dataset's adjacency matrix.

rows or columns ("slicing") and incremental construction of a matrix. After construction, this matrix was converted to compressed sparse row (CSR) format for quick arithmetic operations. To set the transposed elements, the CSR matrix was simply added element-wise to its transpose and thresholded so that non-zero elements were all equal to 1. This highly efficient approach calculated the $1.88 * 10^6 \times 1.88 * 10^6$ adjacency matrix in only 95 seconds. The resulting graph roughly followed a power law (its degree distribution is roughly exponentially distributed) with a mean node degree of $c = 24.3$. Its degree distribution is shown in Figure 1.

However, using this method in practice introduced a perplexing issue in our model. The software package that we utilized for this project, Spektral [6], requires that the rows in the dataset line up with the rows in the provided adjacency matrix, so we were forced to also mergesort our dataset based on time as well. This caused some of our models to take much longer to converge or to not converge at all. We believe that this occurred because Spektral requires that training process the entire dataset in a single batch, with no shuffling between training epochs. This was simply impossible for the GraphSAGE technique as it required using a dense matrix instead of a sparse matrix, so we were forced to split our dataset into batches of size $|V|/16$ even when performing our calculations in the cloud with a sizeable amount of memory. We attempted multiple changes to the dataset to resolve this problem, but the solution ultimately required simply *un*sorting our adjacency matrix after calculating it; the convergence of neural networks can be impeded if the training examples are always passed in a sorted order. This seemingly trivial task was made more difficult by the fact that our matrix was stored in sparse format, but ultimately only added about 1 minute onto our full adjacency matrix calculation time.

3 TECHNICAL APPROACH

3.1 Mathematical Outline

We first define the following variables, as in the documentation of the Spektral Python package [6]:

$G = (V, E)$, all of the information contained in our graph

N = # of nodes

D = # of features per node

X = feature matrix

A = adjacency matrix of the graph

F = # of output features per node

l = index of the current NN layer

W^l = the weight matrix for layer l

b = constant bias term

H^l = feature matrix exiting NN layer l

H_v^l = feature vector exiting NN layer l for node v in G

$N(v)$ = neighbors of node v

$Z = N \times F$ output matrix

3.1.1 The Graph Convolutional Network Approach. Here we provide a brief overview of the mathematical basis of convolutional graph neural networks [8]. Given our defined variables we can express a convolutional NN layer using a non-linear function:

$$H^0 = X$$
$$H^{l+1} = f(H^l, A)$$
$$H^L = Z$$

We now define and parameterize f in such a way that our model can learn what we need it to learn. A naive approach would be to compute this as we do in a Multi-Layer Perceptron (MLP):

$$f(H^l, A) = \hat{A}H^l W^l + b$$

However, this introduces several issues.

(1) By incorporating the adjacency matrix A, we are now summing up all feature vectors of neighboring nodes but ignoring the node that we are looking at. We can simply add the identity matrix to A to resolve this issue. We call this new matrix $\hat{A}$.
(2) A is not normalized so we are going to alter the scale of our feature vectors by including it. We normalize it by multiplying it by the inverse of the diagonal node degree matrix $\hat{D}^{-1}$.
   - Instead of simply normalizing it, we symmetrically normalize it in order to not simply average the neighboring nodes.

We can formally define our new matrices:

$\hat{D}$ = diagonal degree matrix of all nodes V

$\hat{A}$ = adjacency matrix with added self-loops

And express our corrected function as:

$$f(H^l, A) = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^l W^l + b$$

Therefore:

$$H^{l+1} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^l W^l + b)$$

Where σ is a non-linear activation function.

3.1.2 The GraphSAGE Approach. Using our defined variables, GraphSAGE [7] is a closely-related approach, which calculates

$$Z = [\text{AGGREGATE}(X) \| X]W + b$$
$$Z = \frac{Z}{\|Z\|}$$

where AGGREGATE is a generic function applied to aggregate results across the given node's neighborhood. As discussed by the GraphSAGE authors [7], when this aggregation function is the mean, it simplifies into the GCN approach by [8] explained above. GCN can roughly be thought of as a special case, but the more general GraphSAGE allows a popular alternative, a "pooling" aggregator which takes the elementwise maximum of the neighborhood nodes' vector representations. This "max pooling" method of GraphSAGE is what we chose to use in our project.

3.1.3 The GIN Approach. Using the definition provided in the paper from [12] and continuing forward with our defined variables, we parameterize f as a multi-layer perceptron (MLP).

$$f(H_v^l, A) = MLP((1 + \epsilon^l) * H_v^{l-1} + \sum_{u \in N(v)} H_u^{l-1})$$

Briefly, the distinct purpose of the GIN approach (by introducing an MLP) is to allow a non-linear filter. In this way, we effectively nest a neural net inside of our neural net. This allows the GIN to discriminate certain cases, described in detail in the original paper, where the aggregation functions of GCN and GraphSAGE return identical embeddings for nodes with very different graph structures. To summarize broadly, the GIN (one of the most recent approaches in the literature) achieves state-of-the-art performance by increasing the expressive power at each node, at the cost of increasing the size of the parameter space (potentially by a large amount, depending on the structure of the MLP). More formally, the GNN has as much discriminative power as the Weisfeler-Lehman (WL) test, making it "maximally powerful," meaning that it *only maps two nodes to the same location when they have identical subtrees with identical features*, unlike the previous methods [12].

The unnamed hyperparameter ϵ controls the influence of a given node relative to its neighborhood, and while it can be estimated by gradient descent, the original authors did not find this effective; it is better to set it to its default value of 0 [6, 12].

3.2 Implementation

3.2.1 Baseline models. We began our analysis by testing our methods with a baseline deep neural network (DNN) model. This DNN consisted of 3 hidden dense layers with 64 neurons per layer, each followed by ReLU activation and batch normalization. Finally, we utilized an 11-node softmax output layer to output class probabilities. To assess the improvement to prediction accuracy provided by our augmented features, we first trained this model with only the original features, then with only the augmented features, and finally on both. We saw an improvement of 3.2% prediction accuracy from 53.8% to 57% when adding in our augmented features. This underwhelming result can likely be attributed to the fact that NOAA weather stations become very sparse in the western-most quarter of the United States, which could make the elevation and weather less informative than anticipated. For example, a wildfire might be located in an elevated, mountainous region, but a weather station at the base of the mountains will not report the accurate wildfire elevation.

Table 1: Model Validation Categorical Accuracy (%)
(* indicates model not run)
(? indicates model results uninformative)

Model Type	2015	Half	All
Baseline DNN	57.0	*	*
State-of-the-art	*	*	58.0
GCN	53.4	53.7	*
GraphSAGE	65.9	56.0	55.2
GIN	?	*	*
GCN-DNN	52.0	*	*
GraphSAGE-DNN	61.0	*	*
GIN-DNN	43.0	*	*

3.2.2 GNN models. For our GNN models, we chose to train two variants: the GNN on its own and the GNN with our baseline DNN appended onto the end. Since the DNN had far more parameters than the GNN did on its own, this guaranteed that our models were not under-fitting by simply lacking the DNN's number of parameters. These modified "GNN-DNN" models were trained to predict a wildfire's cause not from its feature data, but from *the learned embeddings* at each node. All of our GNNs utilized 5 graph-convolutional layers. We implemented GIN using identical parameters to those by Xu et al. [12] and found no noticeable difference in performance between using 16, 32, and 64 hidden neurons in the convolutional layer's multi-layer perceptron. As such, our results reflect the performance of the 64-node variant. All of our models were optimized using the Adam optimizer and a learning rate of 0.01, based on recommendations included with the Spektral package [6] and Xu et al. [12]; the GIN's learning rate was decayed in accordance with its original authors' work. Our results can be seen in Table 1.

4 EVALUATION

The nodes' classes in this graph (the wildfires' causes) are highly imbalanced: for example, there are more than 400,000 cases of burning debris but under 4,000 cases of structural fire. As a result, it would be optimal to report our results in AUC values. However, the performance of the state-of-the-art method is reported in terms of categorical accuracy, so this was the metric that we employed to compare our models, to each other and to other authors' results.

We began our GNN training process by using a subset of our full dataset, including only wildfires that took place in 2015. The results of this preliminary test already showed GraphSAGE performing at 63% validation-set accuracy, a 6% improvement over our baseline DNN and a 5% improvement over the state-of-the-art.

Given the performance of our GraphSAGE model was much higher than any of our other methods, we first moved forward with running this model on our full dataset. This proved difficult considering GraphSAGE must use dense matrices in Spektral's implementation, however, we were ultimately able to run the model with a batch size of $N/16$. This model was run for 600 epochs at 2

minutes per-epoch and it was found to converge around 55% validation accuracy. This was disappointing, considering the performance of our smaller model, so we decided to compare a few other models to evaluate their performance on larger datasets. GCN could not run on our full dataset due to low-level CUDA errors both locally and in the cloud that prevented us from running our model, with or without modifications to the batch size. As a result, we ran GCN using the first half of our dataset and were able to run it for about 1500 epochs. This model outperformed its 2015-only counterpart, but was still unsatisfactory in comparison to our smaller GraphSAGE model. Finally, we ran a version of GraphSAGE with only half of the data and our batch size set to $N/8$, with the idea that it might exhibit faster training and convergence. After 1500 epochs, this model was performing better than the full dataset model, but was still under-performing in comparison to our 2015-only model.

5 CONCLUSION

This project focused many of its efforts on engineering specific dataset augmentations motivated by a review of published literature and extensive development of an efficient approach to calculating very large adjacency matrices for this wildfire analysis. However, after implementing the model on our full 1.88 million row, 33 column dataset and enlisting these carefully-crafted algorithms, the full-dataset model actually exhibited disappointingly lower performance. We hypothesize that this could be due to the splitting of the adjacency matrix (into several separate, disconnected graphs) that was necessary to fit our training batches within GPU memory. Nonetheless, the 2015 GraphSAGE model still outperformed the state of the art by a full 7.9 percentage points. As a result, it appears that context from surrounding wildfires is key consideration for future research, and our methods represent a notable advancement in wildfire cause prediction.

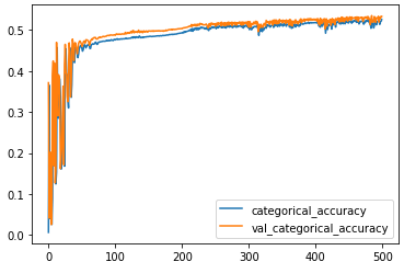

Figure 2: Training and Validation accuracy for GCN training on 2015 data

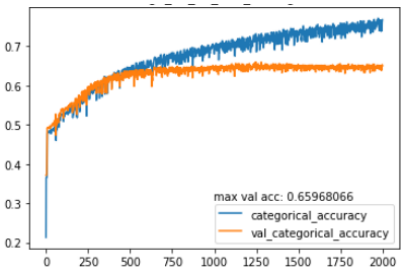

Figure 3: Training and Validation accuracy for SAGE training on 2015 data

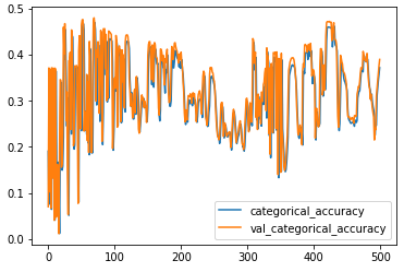

Figure 4: Training and Validation accuracy for GIN training on 2015 data with 64 hidden neurons

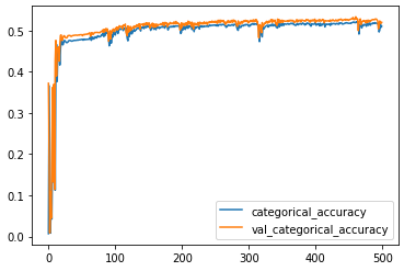

Figure 5: Training and Validation accuracy for GCN-DNN training on 2015 data

REFERENCES

[1] [n.d.]. 1.88 Million US Wildfires. https://kaggle.com/rtatman/188-million-us-wildfires

[2] [n.d.]. NOAA Global Surface Summary of the Day. https://kaggle.com/noaa/noaa-global-surface-summary-of-the-day

[3] [n.d.]. World Cities Database. https://kaggle.com/max-mind/world-cities-database

[4] Jennifer K. Balch, Bethany A. Bradley, John T. Abatzoglou, R. Chelsea Nagy, Emily J. Fusco, and Adam L. Mahood. 2017. Human-started wildfires expand the fire niche across the United States. *Proceedings of the National Academy of Sciences of the United States of America* 114, 11 (March 2017), 2946–2951. https://doi.org/10.1073/pnas.1617394114

[5] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. https://doi.org/10.1145/361002.361007

[6] Daniele Grattarola. 2020. Graph Neural Networks with Keras and Tensorflow. https://github.com/danielegrattarola/spektral original-date: 2019-01-17T11:19:10Z.

[7] William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive Representation Learning on Large Graphs. *arXiv:1706.02216 [cs, stat]* (Sept. 2018). http://arxiv.org/abs/1706.02216 arXiv: 1706.02216.

[8] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]* (Feb. 2017). http://arxiv.org/abs/1609.02907 arXiv: 1609.02907.

[9] National Wildfire Coordinating Group. 2016. Guide to Wildland Fire Origin and Cause Determination. https://www.nwcg.gov/sites/default/files/publications/pms412.pdf

[10] Karen C. Short. [n.d.]. *Spatial wildfire occurrence data for the United States, 1992-2015 [FPA_FOD_20170508] (4th Edition).* Technical Report. https://doi.org/10.2737/RDS-2013-0009.4 type: dataset.

[11] Alexandra D. Syphard and Jon E. Keeley. 2015. Location, timing and extent of wildfire vary by cause of ignition. *International Journal of Wildland Fire* 24, 1 (Feb. 2015), 37–47. https://doi.org/10.1071/WF14024

[12] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks? *arXiv:1810.00826 [cs, stat]* (Feb. 2019). http://arxiv.org/abs/1810.00826 arXiv: 1810.00826.

Table 2: Augmented Dataset Description
(* indicates NA)

Feature	Data Type	Description	Units	Decimal Points
STAT_CAUSE_DESCR	categorical	Wildfire cause categorical indicator	*	*
FIRE_YEAR	int	Year the fire occurred	Years	0
DISCOVERY_DOY	int	Day of year the fire was discovered	Days	0
FIRE_SIZE	int	Estimate of area within the final perimeter of the fire	Acres	0
LATITUDE	float	Latitude of the fire's center	Degrees	6
LONGITUDE	float	Longitude of the fire's center	Degrees	6
STATE	categorical	Two-letter code of the state of the fire's origin	*	*
CONTAINED	bool	Indicator for whether the fire was contained	*	*
DISCOVERY_MONTH	categorical	Month of the year that the fire was discovered	Month	0
DISTANCE_CITY_1000000	float	Distance from the fire to the closest city with population > 1 mil	Degrees	6
DISTANCE_CITY_100000	float	Distance from the fire to the closest city with population > 100 k	Degrees	6
DISTANCE_CITY_10000	float	Distance from the fire to the closest city with population > 10 k	Degrees	6
DISTANCE_CITY_1000	float	Distance from the fire to the closest city with population > 1 k	Degrees	6
APPROX_ELEVATION	float	Elevation of the nearest weather station	Meters	1
DewPoint	float	Dew point of the closest weather station reading	Degrees F	1
Elevation	float	Elevation of the nearest weather station in space and time	Meters	1
Fog	bool	Indicator of fog for the day	*	*
FunnelCloud	bool	Indicator of funnel clouds for the day	*	*
Hail	bool	Indicator of hail for the day	*	*
Rain	bool	Indicator of rain for the day	*	*
Precip	float	amount of rain for the day	Inches	2
Snow	bool	Indicator of snow for the day	*	*
SnowDepth	float	Snow depth	Inches	1
Thunder	bool	Indicator of thunder during for the day	*	*
Visibility	float	Mean visibility for the day	Miles	1
Gust	float	Maximum wind gust reported for the day	Knots	1
Windspeed	float	Mean windspeed reported for the day	Knots	1
MaxWindspeed	float	Maximum windspeed reported for the day	Knots	1
Temp	float	Mean temperature reported for the day	Degrees F	1
MaxTemp	float	Maximum temperature reported for the day	Degrees F	1
MinTemp	float	Minimum temperature reported for the day	Degrees F	1
SeaLevelPressure	float	Mean sea level pressure for the day	Millibars	1
StationPressure	float	Mean station pressure for the day	Millibars	1

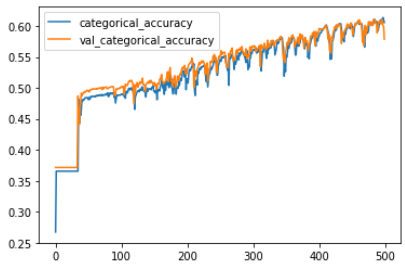

Figure 6: Training and Validation accuracy for SAGE-DNN training on 2015 data

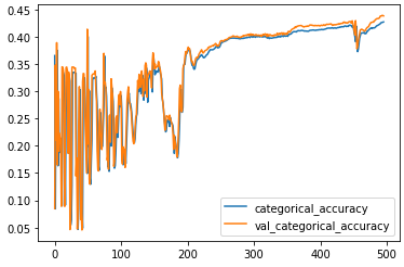

Figure 7: Training and Validation accuracy for GIN-DNN training on 2015 data

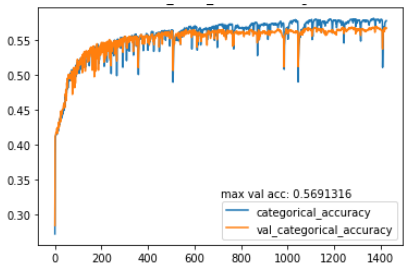

Figure 8: Training and Validation accuracy for GCN training on the first half of the full dataset

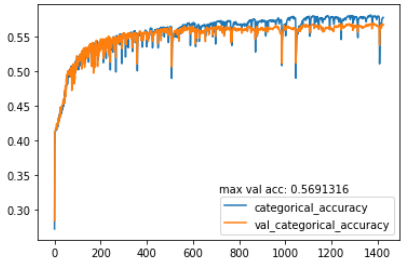

Figure 9: Training and Validation accuracy for SAGE training on the first half of the full dataset

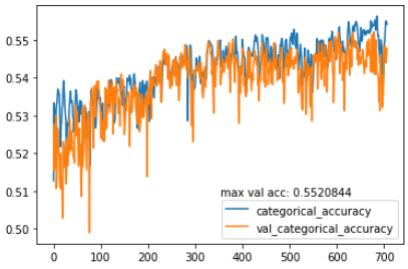

Figure 10: Training and Validation accuracy for SAGE training on the full dataset