

Supplementary Materials

Appendix A. README File

In this Supplementary material, we provide a README file for readers to assist anyone who wishes to run the related computer code of the high-resolution multimodal transformer neural network model described in this paper. The interested readers can write to the corresponding author of this paper requesting the source code of the computer program. The reader should be able to run this code either on your own machine or on Google Colab.

1. Local machine: If running on local machine, please run

```
$ pip install -r requirements
```

or during running it will automatically checking satisfaction for requirements

2. Online Colab: First cell will check the requirements. Colab is available here

Appendix A.1. Requirements

- Python 3.6+
- PyTorch 2.0.0+[59]

Appendix A.2. Usage

The script and data are available from

Data Name	Size
Scripts & Codes	47K
Structured Information Data	6.29M
Imagery Data	1.10G
Pre-trained model	34.2M
FPA_FOD_20170508 Wildfire information	NA
NOAA weather information	NA

Table A.3: Data URL

Appendix A.2.1. Run the code locally

Example command for run a combined model:

```
$ python train.py --with-vegetation=True\  
                  --satellite-img=True\  
                  --gray-scale=True\  
                  --resample-method='undersampling'\  
                  --archive=True\  
                  --model-selection='hybrid_model'
```

```

—learning-rate=0.01\
—epochs=100\
—display_step=10\
—batch=True\
—batch-size=64

```

This command in terminal will allow training the model vegetation and image features using undersampling method. The model is hybrid model with 64 mini-batches. It will run 100 epochs with learning rate 0.01, for every 10 steps, and then it will display the loss or accuracy. After finishing training, the model will be saved to './saved_progress'.

You can also run it with

```
$ python train.py -h
```

to see help and all available arguments.

Or using UI as following by running

```
$ python wildfire_ui.py
```

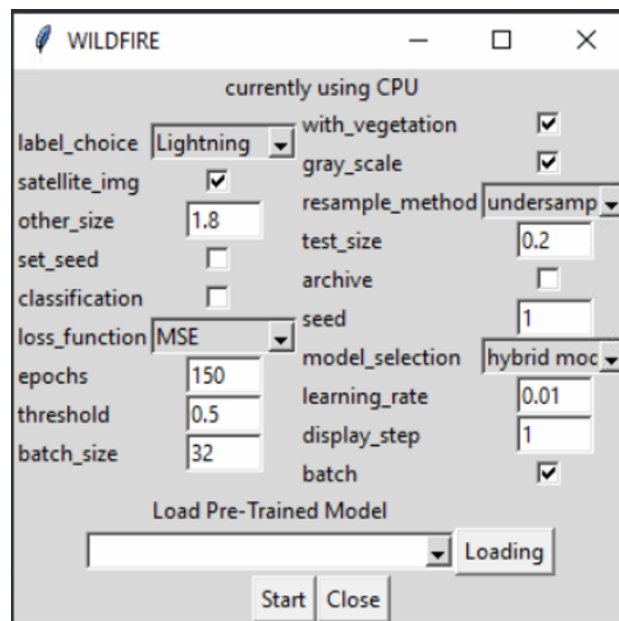


Figure A.24: User Interface

This UI will not run the program, instead it will generate a command state in Appendix Appendix A.2.1.

Appendix A.2.2. Instructions on Colab

First time run steps:

1. Loading Required Package
2. Create Symbolic Link

3. Util Functions
4. Parameter Setting
5. Read Subset Data & Create DataFrame
6. Create Dataset (Train set & Test set)
7. Neural Network (or F-CNN & ResNet)*
8. Hybrid Model
9. Initialize model
10. Training

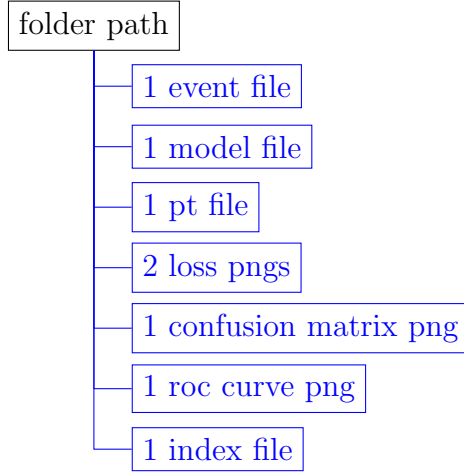
- Repeat training:
 - Training (step 10)
- Re-train a new model with same data
 - Initialize model (step 9)
 - Training (step 10)
- Train a new model with new shuffle data
 - Create Dataset (Train set & Test set) (step 6)
 - Initialize model (step 9)
 - Training (step 10)
- Train a new model with some arguments change
 - Parameter Setting (step 4)
 - Read Subset Data & Create DataFrame (step 5)**
 - Create Dataset (Train set & Test set) (step 6)**
 - Initialize model (step 9)
 - Training (step 10)
- Results
 1. Accuracy
 2. Confusion matrix
 3. Tensorboard
- Load pre-trained model***

```
import torch
model = torch.load(<desired model path>)
```

Here the symbol * indicates it applying to any desired models

The symbol ** indicates it applying when change input data (i.e test size, undersampling)

The symbol *** indicates that the saved progress may include 8 files



Event file stores the log information for training and test, model and .pt files store the model and all parameters separately, loss PNGs are the plots for the loss based on epochs or batch steps, confusion matrix png file displays the confusion matrix as a table, roc curve png show the ROC curve for the confusion matrix, and index file contains the random indices when data was shuffled and split for training and test initially for reuse purpose.

Params	Baseline	Hybrid
w/ veg	True	True
udersample	≥ 1.5	≥ 1.5
test size	≥ 0.15	≥ 0.15
learning rate	0.01	0.01
epochs	≤ 300	≤ 50

Table A.4: Recommended Tuning

Appendix A.3. Parameter Descriptions

Parameters	Descriptions	Type
connect to google drive	allowing script connect to user's Google Drive	boolean
download_pre_trained	download pre-trained model	boolean
label choice	choose desired label	string
with vegetation	check box for vegetation categories as features	boolean
satellite images	check box for satellite images as features	boolean
resample method	choose resample methods	string
other size	with previous checked, choosing proportion for undersampling	float
test size	the proportion of test set	float
archive	check box for save the results	boolean
set seed	check box for control the randomness	boolean
seed	the seed for randomness	integer
model selection	select training models	string
loss function	choose loss function	string
learning rate	learning rate for optimizer	float
epochs	the amount of iterations training will run	integer
display steps	show the loss and accuracy	integer
threshold	a number larger than it will be false otherwise true	float
batch	check box for allowing batch during training	boolean
batch size	with previous checked, the size of each batch	integer

Table A.5: Parameter Settings Explanations

Appendix A.4. Hyperparameters for WIIN

Category	Parameter	Value
Dataset Settings	Label Choice	Lightning
	Undersampling	True
	Class Ratio (other_size)	1.8
	Use Vegetation	True
	Use Satellite Image	False
	Image Channels	3
	Image Size	100
Data Split	Test Size	0.2
	Random Seed Enabled	False
	Seed Value	1
Model Selection	Model Type	Deep Neural Network
	Alternative Model	Deep NN + CNN
Training Parameters	Loss Function	MSE
	Learning Rate	0.01
	Epochs	100
	Display Step	5
	Threshold	0.5
	Batch Training	False
	Batch Size	128

Table A.6: Model and Dataset Configuration Parameters

Appendix B. Google Earth Pro Settings

Parameters	Value
Scale	1:1000
Historic imagery	on
Terrain	on
Cloud	off
Atmosphere	off
Sun	off

Table B.7: Settings for Google Earth Pro

Appendix C. Summary of features utilized in data

In this Section, we provide the summary of features utilized in the dataset which will feed into the model

features	source	amount
latitude, logitude	FPA_FOD	2
temperature for 7, 15, and 30 days average	FPA_FOD	3
Wind for 7, 15, and 30 days average	FPA_FOD	3
humidity 7, 15, and 30 days average	FPA_FOD	3
precipitation 7, 15, and 30 days average	FPA_FOD	3
vegetation category	NCEI	7
imagery data	Google Earth	100×100

Table C.8: Summary of features utilized in data

Appendix D. Data Content Description

In this Section, we provide the information data input format of this study, which is used as the data input format. They are given as follows.

FOD_ID	Global unique identifier
FIRE_NAME	Name of the incident, from the fire report (primary) or ICS-209 report (secondary).
date(julianday(DISCOVERY_DATE))	Date on which the fire was discovered or confirmed to exist
DISCOVERY_TIME	Time of day that the fire was discovered or confirmed to exist
STAT_CAUSE_CODE	Code for the (statistical) cause of the fire
STAT_CAUSE_DESCR	Description of the (statistical) cause of the fire
LATITUDE	Latitude (NAD83) for point location of the fire (decimal degrees)
LONGITUDE	Longitude (NAD83) for point location of the fire (decimal degrees).
FIRE_SIZE_CLASS	Code for fire size based on the number of acres within the final fire perimeter expenditures
FIRE_SIZE	Estimate of acres within the final perimeter of the fire

Table D.9: FPA_FOD description[20]

Appendix E. General Wildfire Causes defined in National Wildfire Coordinating Group

In this Section, we list all known or defined causes of a wildfire in National Wildfire Coordinating Group (NWCG).

Existing NASF Cause Code	Proposed Fire Cause Standard
1-Lightning	Natural
2-Equipment Use	Equipment and vehicle use
3-Smoking	Smoking
4-Campfire	Recreation and ceremony
5-Debris Burning	Debris and open burning
6-Railroad	Railroad operations and maintenance
7-Arson	Arson
8-Children	Misuse of fire by a minor
9-Miscellaneous	Other causes
10-Fireworks	Fireworks
11- Power line	Power generation/transmission/distribution
12-Structure	Structure (under Other)
	Firearms and explosive use
	Undetermined (Human or Natural)

Table E.10: General Causes by National Wildfire Coordinating Group[60]

Appendix F. Vegetation Categories

In this Section, we provide a list of vegetation categories that are used in the present study, which is given as follow.

Vegetation	Dominant vegetation in the areas
1	Tropical Evergreen Broadleaf Forest
2	Tropical Deciduous Broadleaf Forest
3	Temperate Evergreen Broadleaf Forest
4	Temperate Evergreen Needleleaf Forest TmpENF
5	Temperate Deciduous Broadleaf Forest
6	Boreal Evergreen Needleleaf Forest
7	Boreal Deciduous Needleleaf Forest
8	Savanna
9	C3 Grassland/Steppe
10	C4 Grassland/Steppe
11	Dense Shrubland
12	Open Shrubland
13	Tundra Tundra
14	Desert
15	Polar Desert/Rock/Ice
16	Secondary Tropical Evergreen Broadleaf Forest
17	Secondary Tropical Deciduous Broadleaf Forest
18	Secondary Temperate Evergreen Broadleaf Forest
19	Secondary Temperate Evergreen Needleleaf Forest
20	Secondary Temperate Deciduous Broadleaf Forest
21	Secondary Boreal Evergreen Needleleaf Forest
22	Secondary Boreal Deciduous Needleleaf Forest
23	Water/Rivers Water
24	C3 Cropland
25	C4 Cropland
26	C3 Pastureland
27	C4 Pastureland, 28:Urban land

Table F.11: Vegetation Categories

Appendix G. Precision

In this Section, based on the conditional probability of a random event, we provide a mathematical formula to calculate the precision of the developed high-resolution Bayesian learning neural network model, which is given as follows.

$$\begin{aligned}
Precision &= \mathbb{P}(actual = 1 | predict = 1) \\
&= \frac{\mathbb{P}(actual = 1, predict = 1)}{\mathbb{P}(predict = 1)} \\
&= \frac{\mathbb{P}(predict = 1 | actual = 1) \mathbb{P}(actual = 1)}{\mathbb{P}(predict = 1, actual = 1) + \mathbb{P}(predict = 1, actual = 0)} \\
&= \frac{\mathbb{P}(predict = 1 | actual = 1) \mathbb{P}(actual = 1)}{\mathbb{P}(predict = 1 | actual = 1) \mathbb{P}(actual = 1) + \mathbb{P}(predict = 1 | actual = 0) \mathbb{P}(actual = 0)} \\
&= \frac{TPR \cdot n}{TPR \cdot n + FPR \cdot (1 - n)}
\end{aligned}$$

where n is the proportion of the actual true. Based on the equation above, precision is related to the proportion of true and false data, once the $\frac{FPR}{TPR}$ is fixed, it increases with n increases. However, TPR and FPR are not related to the proportion of the data, since $TPR = \mathbb{P}(predict = 1 | actual = 1)$ and $FPR = \mathbb{P}(predict = 1 | actual = 0)$ [56].

Appendix H. Settings of Baseline Model

Parameters	Values
Hidden Layers	[256, 128, 64, 32, 4]
Loss Function	Mean Square Error (Cross Entropy)
Optimizer	Adam optimizer[50]
Learning Rate	0.01
Weight Decay	1e-4
Epochs	100

Table H.12: Hyperparameters for Neural Network

Appendix I. Residual Network

At this stage, feature maps from the original captured image are extracted so that the model can both maintain its performance and save computation power. As shown from Figure 9, the mini-batched images with the size of (100, 100) will first go through a convolutional block, which includes a convolutional layer, a batch normalization layer, an activation layer, and a max-pooling layer, as shown on the left of Figure I.25 (a). The convolutional layers are used to extract features from any previous procedures, and it will output a new feature map of the previous stage of an image or feature map [53]. Mathematically, for a grey-scaled image $I \in \mathbb{R}^{W \times H \times 1}$, where W stands for the width of the image, and H is the height of the image, the ij -th entry of output feature map by using the kernel $K \in \mathbb{R}^{p \times q}$ can be represented as [61]

$$\sum_i \sum_j K_{(i,j)} I_{(p-i, q-j)}$$

In this stage, the convolutional layer has a kernel size of 5, stride of 2, and padding of 3. Taking Figure I.25 (b) as an example, the blackboxes with 0s are the padding area, and the rest are pixels of the original image. The 5×5 yellow and green shaded areas will multiply weighted kernels that will be trained at different times, and each time, the kernel will move two boxes since the stride is set to 2.

Similar to the convolutional layer, the maxpooling layer will keep the main features and reduce parameters in the model, it removes all the values except for the maximum value in each filter process [53]. It will then obtain the maximum numbers within each kernel instead of multiplying the trained weights. Therefore, after the convolutional block, the image size will become (51, 51) as unnecessary features would be removed.

Then, the convoluted images will be fed into two residual block with 3 layers and 2 layer separately. The residual blocks are mainly two convolutional blocks without the maxpooling layer but add with downsampled, if necessary, initial inputs as residual. As shown in right of Figure I.25 (a), for a single residual block, the new residual $F(x)$ will add back to initial input value x to create a new output $F(x) + x$. With the help of adding residuals to the output, ResNet can provide a good feature map from original image without losing too much information and prevent gradient vanish or explode [62, 40]. Even the fed data have a small size of image, it can still work well. After two residual blocks, the image size reduce to (13, 13), then continue to feed the processed images into the last maxpooling layer. Mathematically, let $F(\cdot) = BN(Conv(\cdot))$

$$\begin{aligned} z_0 &= F(x) \\ z_0 &= \text{maxpool}(g(z_0)) \\ z'_l &= F(g(z_{l-1})) \\ z_l &= g(F(z'_l) + z_{l-1}) \\ z_l &= z_l + \sum_{i=1}^{L-1} F(z_i) \\ y &= \text{maxpool}(z_l) \end{aligned}$$

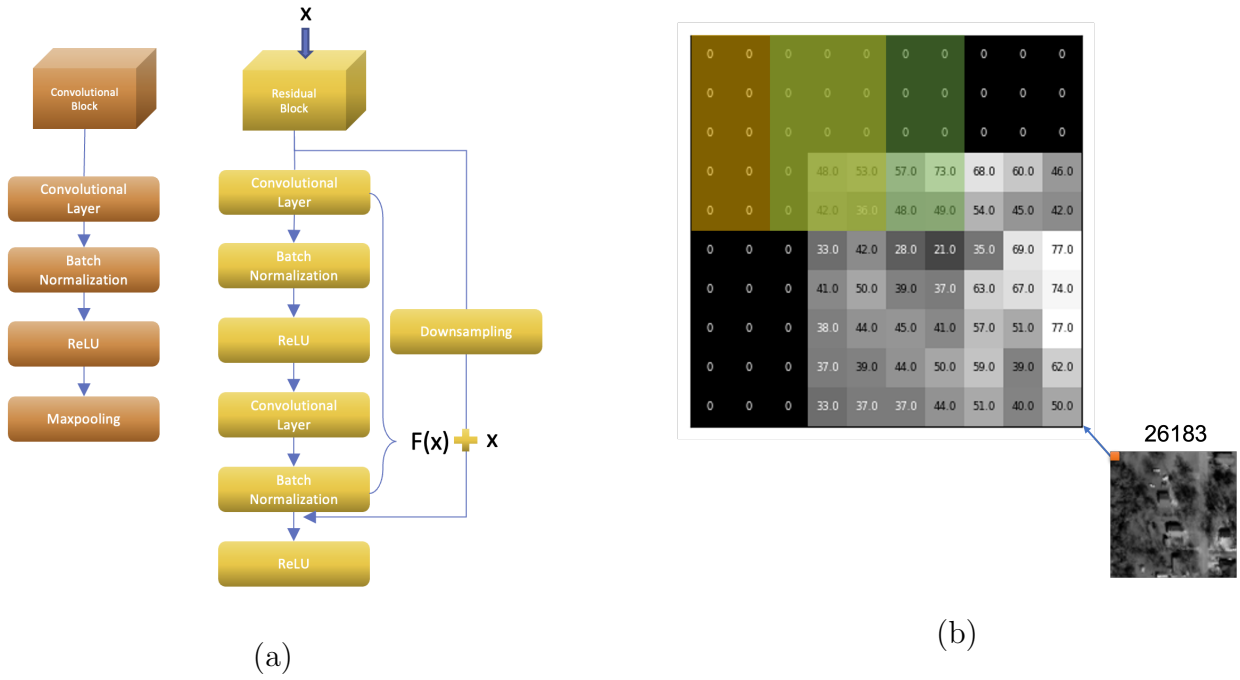


Figure I.25: (a) Convolutional Block; RIGHT: Residual Block, and (b) Top left corner of image 26183 in the first convolutional layer with kernel 5×5 , stride 2, padding 3.

where BN stands for the batch normalization layer, $Conv$ is the convolutional layer, $maxpool$ is the maxpooling layer, and $g(\cdot)$ is the activation function.

Parameters	Values
Layer(Kernel, Stride, Padding)	conv(5,2,3); maxpool(3,2,1); conv(3,1,1)+conv(3,1,1) $\times$ 3; conv(3,2,1)+conv(3,2,1) $\times$ 2; maxpool(4,1,0)
Epochs	300
Batch Size	32

Table I.13: Hyperparameters for ResNet

Appendix J. Wildfire Images Transformer (WIT)

In the transformer stage, similar to a vision transformer (ViT) (See: [42]), the output image from the previous stage will be fed into a convoluted embedded patch layer so that the image features will be flattened, as shown in Figure J.26, meanwhile, a learnable feature token will be appended to the newly flattened features, as shown in the left of Figure J.27. Mathematically, this block with the output shown in Figure J.26 can be represented as

$$\begin{aligned} z_0 &= [x_{token}^0; x_{patch}^1 E; x_{patch}^2 E; \dots; x_{patch}^N E] + E_{position} \\ z'_l &= MSA(LN(z_{l-1})) + z_{l-1} \\ z_l &= MLP(LN(z'_l)) + z'_l \\ y_{image} &= Linear(Conv(z_l^{1:N})) \\ y_{token} &= Linear(LN(z_l^0)) \end{aligned}$$

where x_{patch}^i is the patches, N is number of patches, which can be calculated as $\frac{H \times W}{(patch\ size(P))^2}$. E_s are the linear projection of flattened patches, a fully connected layer, which will map the learned features onto sample space (See: [53]), E_{pos} is the positional encoder, MSA is the multi-head self-attention layer, LN is the linear normalization layer, MLP is the multi-layer perception layer, *linear* is the fully connected linear layer, and *conv* is the convolutional layer.

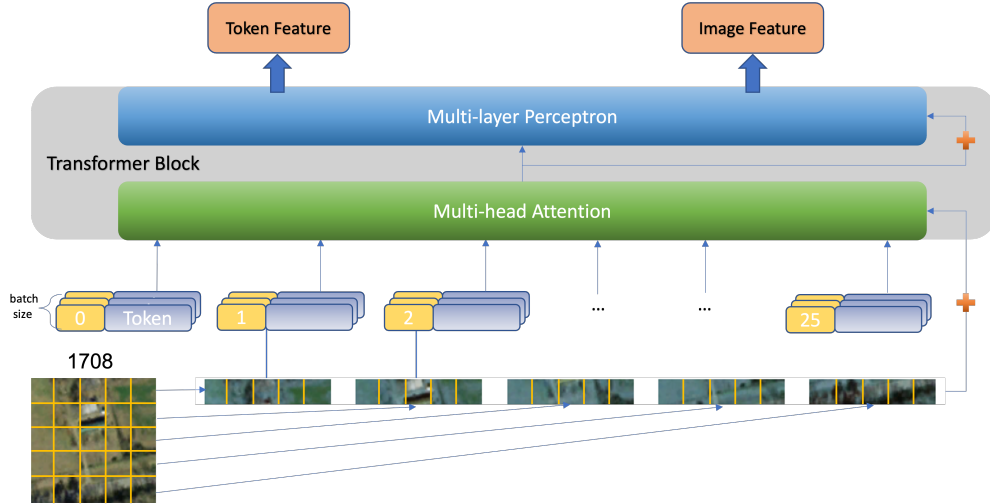


Figure J.26: Vision Transformer

For instance, for an image in the output of ResNet, $x_i \in \mathbb{R}^{1 \times 10 \times 10}$, with the patch size of 2, the flattened vector should have $\frac{H \times W}{P^2} = \frac{100}{4} = 25$ patches concatenate an additional token vector and positional encoding. The output dimension will become a $x \in \mathbb{R}^{(25+1) \times 4}$ features. Our model applied a convolution method to the image since our patch size is small, only 4, with the output channel 64. Thus, the output dimension of the embedded patch layer is (26, 64).

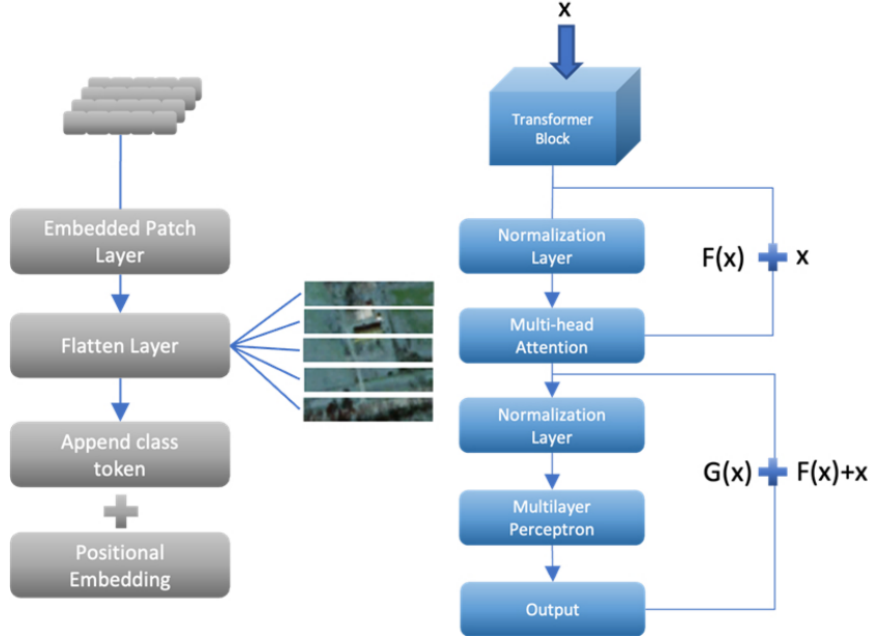


Figure J.27: LEFT: Embedded Patch Layer; RIGHT: Transformer Block

Then, the data will have two transformer blocks, each of which will have a multi-head attention layer and a multilayer perceptron, as shown in the right of Figure J.27. In the multi-head attention layer, the flattened input feature vectors with appending tokens will apply linear projections by multiplying three different weight, $w_i \in \mathbb{R}^{P^2C \times \frac{P^2C}{n_head}}, i \in \{q, k, v\}$ to three new vectors $Q(query)$, $K(key)$, and $V(value)$, for instance $w_q x = Q$. Thus, in this case, Query, key, and value vectors should have shape $(b, 26, 8, 8)$ as the hidden dimension is $64(8 \times 8)$. The n_head stands for the number of heads defined for multi-head attention. Vaswani et al. [41]. define the output of the attention layer as concatenating the various heads and then multiplying by a matrix w^o .

$$\text{concat}(head_1, \dots, head_h)w^o = head_1w_1^o + \dots + head_hw_h^o$$

where $w^o = \begin{bmatrix} w_1^o \\ \vdots \\ w_h^o \end{bmatrix}$ As the results are concatenated, the dimension is $\frac{P^2C}{n_head} \times n_head = P^2C$.

Thus, at the end of this layer, the dimension is still $\mathbb{R}^{P^2C \times P^2C}$. Once Q, K, V are obtained, the score could be computed by $\text{softmax}(QK^T)$, and logit are scaled by $\frac{1}{\sqrt{d_{kv}}}$ for the scaled dot-product score as follows,

$$Z = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{kv}}}\right)V,$$

where Z is scaled dot-product attention [41].

Thus, at the end of this layer, the dimension is still $\mathbb{R}^{P^2C \times P^2C}$. Once Q, K, V are obtained, the score could be computed by $\text{softmax}(QK^T)$, and logit are scaled by $\frac{1}{\sqrt{d_{qv}}}$ for the scaled dot-product score as follows,

$$Z = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{qv}}}\right)V,$$

where Z is scaled dot-product attention [41].

During the multi-layer perception stage, the input will flow into a fully connected linear layer with activation function Gaussian Error Linear Units (GELUs) [63], which is defined as

$$\begin{aligned} \text{GELU}(x) &= xP(X \leq x) \\ &= x \cdot \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{x}}\right)\right] \end{aligned}$$

and a dropout layer to prevent overfitting. In the last linear layer, the output dimension will be changed back to its origin. Thus, the dimension of the vectors has no changes.

Instead of outputting a token class only like ViT, our model will also output image features, which can be treated as summarized features of an image. Thus, at the output stage of the wildfire image information network, we will have extra features for appending to the neural network model.

Parameters	Values
Number of Layers	2
Patch Size	2
Number of Head	8
Multi-Layer Perceptron Neurons	8
Dropout Rate	0.2

Table J.14: Hyperparameters for Transformer

Appendix K. Example of Sample Outputs

In this Section, we show an example of sample outputs. Figure K.28 displays the sample outputs of the image No. 1333.

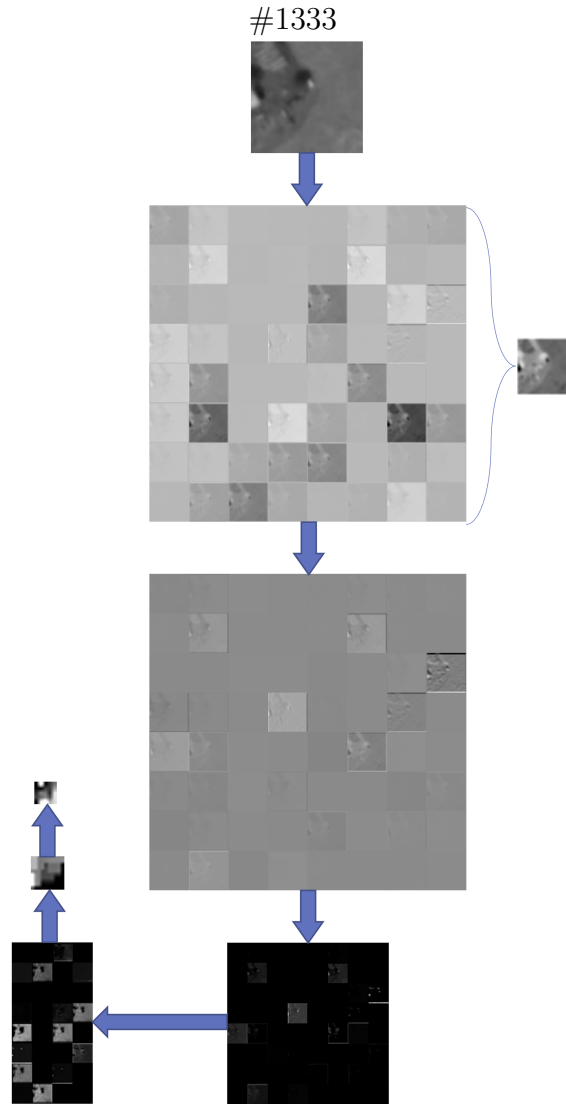


Figure K.28: Image flow of image #1333 for each layer. From top: original input → 1st convolution output → 1st batch normalization output → 1st maxpooling output → 1st residual block output → WIT output