Generative Modeling for Climate Risk Assessment in Mortgage Portfolios

Jonathan C Hague
jhague@stanford.edu

Abstract

Climate scenario analysis is the latest modeling tool financial regulators in major economies are using to help grasp the effect of climate change on the health of the financial system. Those exercises rely primarily on sensed data and trained climate models that apply a risk output to loan portfolio geolocations. Financial institutions in turn take those risk coefficients and monetize them to derive and deliver financial disclosures to users of financial information. Wildfire is one major climate hazard that is assessed in scenario analysis. We explore the possibility of applying deep learning methods to derive Logistic Regression (Logit) classification of satellite images indicating prevalent risk of wildfire based on images geolocation and the latent space of a trained Variational Autoencoder (VAE) model. The results show that a VAE-Logit architecture is a promising framework for feature extraction and classification of climate hazard satellite images for the assessment of wildfire risks in geographical areas based on satellite images and their respective geolocation[1].

1 Introduction, Motivation, and Related Work

1.1 Introduction

Climate Risk is an emerging theme in the financial industry. Regulators all over the world are racing to draft, propose, and implement supervisory measures to ensure Climate Risk is incorporated in the overall assessment and disclosure of risks by firms in the market. The latest efforts in that direction is the The Federal Deposit Insurance Corporation (FDIC) cross agency principles issued in October 2023 titled "Principles for Climate-Related Financial Risk Management for Large Financial Institutions" (`https://www.fdic.gov/news/financial-institution-letters/2023/fil23056.html`). Another major effort is that of the US Securities and Exchange Commission (SEC) proposed rules to enhance and standardize climate-related disclosures for investors (`https://www.sec.gov/news/press-release/2022-46`). This effort is in line with that by regulators in the EU, UK, Canada, and Japan (`https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv%3AOJ.L_.2022.322.01.0015.01.ENG&toc=OJ%3AL%3A2022%3A322%3ATOC`).

Consequently, major financial institutions in the US and around the world are now faced with a flood of new (climate) data streams to capture, model, analyze and disclose. The Federal Reserve Board (FRB) Climate Risk Scenario Analysis pilot exercise was completed in July 2023 and included the US systemically important financial institution (SIFI). The purpose of climate scenario analysis is to assess the financial system's exposure to climate risk (physical risk emanating from climate hazards, and transition risk emanating from changes in policies), and the capabilities of the system to adapt to various climate transition scenarios under globally defined major socioeconomic pathways.

One main objective of those exercises is to look at commercial banks mortgage portfolio and derive, through modeling, a value for the expected losses on loans and collateral over the life of the loan due

[1] Github repository with this work's code: `https://github.com/Jonathan-Hague`

Preprint. Under review.

to climate risk, under various climate assumptions. The asset geolocation is used to derive that risk of loss metric.

In this paper, we are working with open source data feeding mainstream climate risk models used by major US banks on their Canadian real estate portfolio with focus on one climate risk hazard mandated by regulators: wildfire.

Our objective is to find an architecture that is interpretable and relatively less resource intensive to utilize. For that purpose, we built an architecture using VAE and Logit to assess wildfire risk in geographical areas. The model is trained on satellite images of regions marked by their longitude and latitude coordinates, specifically focusing on areas with a history of wildfires covering more than 0.01 acres in Quebec province of Canada.

1.2 Motivation

Climate Scenario Analysis as defined by the Federal Reserve are used *"to learn about large banking organizations climate risk-management practices and challenges and to enhance the ability of both large banking organizations and supervisors to identify, measure, monitor, and manage climate-related financial risks"*`https://www.federalreserve.gov/publications/climate-scenario-analysis-exercise-instructions.htm`. These are similar modeling exercises to stress testing that financial institutions are required to perform on regular basis to assess their capital adequacy under various high risk scenarios.

Hence, financial institutions are in a race to take physical and transition climate risk data collected and modeled by environmental companies and agencies and use those data-sets in their production pipelines for measurement of Climate Risk related to their loans and investment portfolios. Data quality and integrity of data elements (open source or proprietary) involved in this process must be tested and documented throughout various capture-transport-use frameworks. Our motivation is to seek a method to evaluate the accuracy of one of those datasets used on climate risk assessment for assets in Quebec, Canda.

1.3 Related Work

Our literature review focused on climate related application of deep learning methods, particularly around the use of sensed data.

A major paper that has been an inspiration of this work is a paper published in 2020 titled *"Anomaly detection based on machine learning in IoT-based vertical plant wall for indoor climate control"* (`https://www.sciencedirect.com/science/article/pii/S0360132320305837`).

2 Approach and Method

2.1 Data Processing

We will be analyzing a sample of 2,000 satellite images from the Quebec province of Canada (`https://open.canada.ca/data/en/dataset/9d8f219c-4df0-4481-926f-8a2a532ca003`).

The source shows forest fire map indicating forest fires that occurred mainly in southern Quebec. Satellite images of these locations are then obtained using the MapBox API, ensuring high-resolution and up-to-date imagery. Each image geolocation is embedded to the image in its file name.

The dataset is split into training, testing and validation sets following a typical distribution: 70% for training, 15% for testing, and 15% for validation. The images are reprocessed to conform to a uniform size (350x350 pixels) suitable to our VAE model.

We defined a custom dataset class named GeoTaggedImageDataset, which inherits from PyTorch's Dataset class. This class is specifically designed to handle a dataset of geotagged images, which are images that have associated geographical location data (latitude and longitude). The data retrieval method is used to retrieve an image and its corresponding geolocation data based on an index (idx). It loads an image from its path using "read_image". The image is then resized to 350x350 pixels using "TF.resize". The pixel values are then normalized to the range [0, 1]. The filename of the image,

which contains the latitude and longitude information, is parsed to extract these values which are then converted to a tensor.

Our objective is to predicting the likelihood of wildfires in geolocated satellite imagery.

2.2 VAE Model

As mentioned above, the input of the VAE model are satellite images (350*350 pixels) of wildfire and no-wildfire with embedded geolocation of each image in its file name. The output is a reconstructed image of same dimensions.

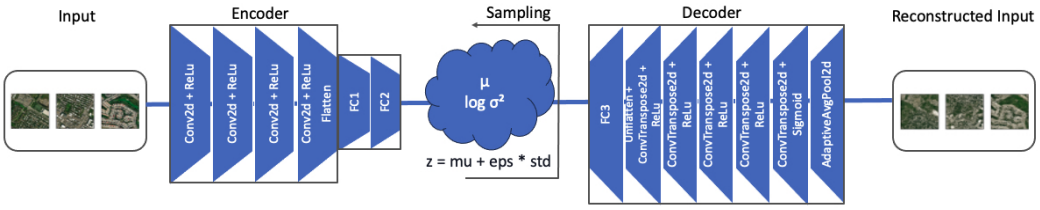

Figure 1: Variational Autoencoder Diagram

Encoder

The encoder objective is to compress the input image into a lower-dimensional latent space. It contains four convolutional layers that apply a number of filters or kernels to the input image creating feature maps. The parameters we used (3, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1)) show that those layers take an input with 3 channels (RGB image), apply 32 different filters, each of size 4x4, with a stride of 2 (how far the filter moves each time), and a padding of 1 (border of zeros around the input to keep the feature map size consistent).

Convolutional layers are particularly effective for image data as they are designed to detect local patterns mainly edges, textures, and colors in an image. They can capture spatial hierarchies in the data, where higher-level layers can recognize more complex structures built from the simpler patterns recognized by lower-level layers. These networks are robust to variations in the location of features in an image. This is crucial for dataset of satellite images we are using in this paper as same features (edge or texture) can appear in various parts of the image.

A Rectified Linear Unit (ReLU) activation function is applied after each convolutional layer to introduce non-linearity. This allows the network to learn more complex patterns. ReLU makes the network less dense and reduces the risk of overfitting.

Together, the convolutional layers and ReLu allow the VAE to efficiently learn and represent the essential features in the input images. This is very importnat for the compression into the latent space and the subsequent reconstruction in the decoder.

The last convolutional layer and ReLu is followed by a flatten operation that converts the 2D feature maps into a 1D vector that can be fed into the first fully connected layers. It converts a multi-dimensional tensor into a one-dimensional tensor (a vector). It preserves the batch size (the first dimension) and combines all the remaining dimensions of the input tensor into the single second dimension.

Lastly, the encoder contains two fully connected layers that have 112,896 (256*21*21) input features, coming from the flattened convolutional layers, and project them down to 512 features. FC1 computes μ; the mean of the latent space distribution for each input. FC2 computes the log-variance ($\log \sigma^2$) of the latent space distribution. The latent variables objective is to capture the key information needed to represent the distribution of the input data.

Reparameterization: VAE sampling

The reparameterization technique, known as the "reparameterization trick", allows the gradient of the loss function to be backpropagated through the random sampling step. By reparameterizing the sampling, the model can be trained from end to end using backpropagation.

The method returns ($z = \mu + \epsilon * \sigma$), which is a sampled point from the latent space distribution. By adding the random noise and scaled by the standard deviation to the mean, we effectively sample from the distribution represented by (μ) and log σ^2 (logvar).

The method calculates the standard deviation (σ) of the latent space distribution by taking the exponential of half the log-variance: σ = torch.exp(0.5 * logvar). The 0.5 factor is used because the standard deviation is the square root of the variance, and taking the exponential of half the log-variance effectively undoes the log and square root operations.

A random noise vector ϵ is sampled using torch.randn_like(std) that generates a tensor of random numbers from a standard normal distribution (mean = 0, standard deviation = 1) that has the same shape as σ. This random sampling step is crucial because it introduces stochasticity into the model, which is a defining characteristic of any VAE. The output of the reparametrization step is a latent vector z of size [batch_size, 512].

Decoder

The decoder starts with the fully connected layer FC3 that serves as a bridge between the low-dimensional latent space and the high-dimensional space required for the transposed convolutional layers in the decoder part of the VAE. It is a crucial component that enables the VAE to reconstruct the original input images from their latent representations.

The role of FC3 is to take the compressed representation (latent vector z), and project it back to the size that is required for the input of the first transposed convolutional layer in the decoder. It basically 'upscales' the latent features to a higher-dimensional space that can be reshaped into a 3D tensor compatible with convolutional operations. It takes the 512-dimensional latent vector and projects it back to the original flattened size, preparing it to be reshaped and processed by the decoder.

This step is followed by an unflatten operation that converts the 1D vector back into a 3D volume with shape (256, 21, 21), which is the expected input shape for the transposed convolutional layers. It performs a linear transformation (using a weight matrix and a bias vector) from the latent space to a larger vector space. FC3 transforms the 512-dimensional vector from the latent space to a vector of size 112,896 which is specifically chosen to match the number of input features needed for the convolutional layers of the decoder.

The following five layers are the inverse of the convolutional layers. They 'upsample' the feature maps to larger spatial dimensions. For example, (256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1)) indicates upsampling the 256-channel feature map to a 128-channel map, doubling its width and height. Each is followed by a ReLu activation function to introduce non-linearity similar to the encoder.

A sigmoid activation function is applied after the last transposed convolutional layer. It is applied to each pixel in order to scale the output between 0 and 1, which is necessary for reconstructing the input image since image pixel values were normalized.

The last layer of the decoder adaptively pools the feature map to produce a fixed-size output, which matches the original input image size (350x350 image).

Loss Function

We defined the reconstruction loss function as the sum of the Binary Cross Entropy (BCE) and Kullback-Leibler Divergence (D_{KL}). This combination helps the VAE reconstruct the data accurately while regularizing the latent space.

We chose BCE because we are dealing with binary or bounded data of normalized images between 0 and 1. BCE here treats each pixel as an independent Bernoulli distribution. Basically requiring our model to process the reconstruction of each pixel as a binary classification problem.

In essence, BCE measures the difference between the reconstructed images (recon_x) and the original images (x). While D_{KL} directs the latent space to follow a standard normal distribution.

We used the Adam optimizer to dynamically adjust learning rates (1e-3) based on the model's individual weights during training.

Our loss L function is defined as below:

$$L = BCE + D_{KL}$$

Where:

$$BCE = -\sum[x \cdot \log(\hat{x}) + (1 - x) \cdot \log(1 - \hat{x})]$$
$$D_{KL} = -\frac{1}{2}\sum[1 + \log(\sigma^2) - \mu^2 - \sigma^2]$$

Where:

x represents the original data points.

$\hat{x}$ represents the reconstructed data points by the VAE.

μ represents the mean of the latent variable distribution.

σ^2 represents the variance of the latent variable distribution.

2.3 The Logit Model

Next, we are using a logistic regression model to classify into wildfire and no-wildfire the data that has been featured from the above studied Variational Autoencoder (VAE), in addition to the geolocation information of each image.

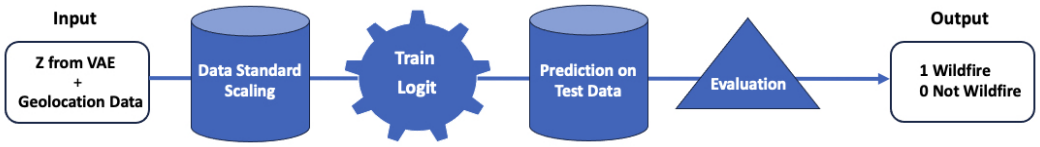

Figure 2: Logistic Regression Diagram

We used the StandardScaler class from Scikit-learn to standardize the features and scaling to unit variance and created a logistic regression model with a high number of maximum iterations to allow convergence.

We trained the logistic regression model using the scaled training data and corresponding labels (wildfire, no-wildfire).

3 Results, Visual Inspection, and Benchmark

3.1 Results

Training and testing datasets are processed using the GeoTaggedImageDataset class. DataLoader objects are created for both datasets to efficiently load the data in batches. The VAE model is trained and tested for 10 epochs (num_epochs = 10), allowing the model to iteratively learn from the training data and then assess its performance on the testing data.

In the forward path, The VAE processes the data to get the reconstructed images and latent variables. In the training function, the loss is computed and gradients are back propagated through the network. The optimizer updates the model parameters. Finally, The loss for each batch is accumulated to calculate the average loss for the epoch which we noticed decreased between the first and last epoch.

The Logit model is trainted and tested, then performance evaluated on the validation data set. We notice an acceptable outcome in the confusion matrix with a precision of 0.97 as shown below and an AUC of 0.9978. The model generates an accuracy of 0.9747 which is significant.

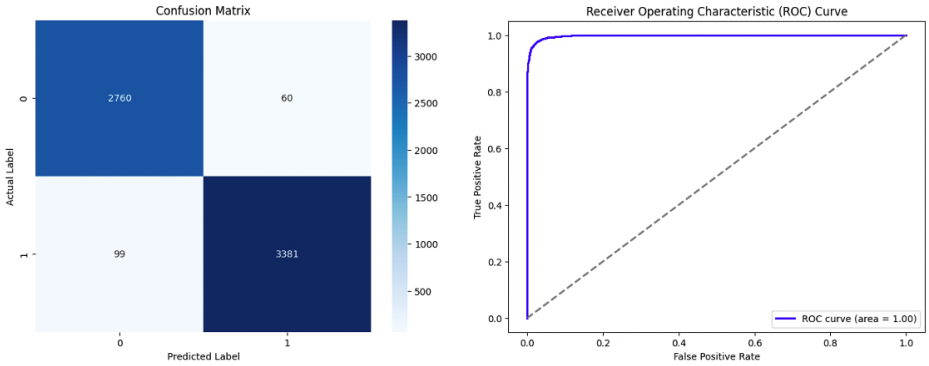

Figure 3: Logit Evaluation Confusion Matrix and ROC Curve

3.2 Visual Inspection of VAE

Visual inspection of the latent space and the generation process helps us assess the quality of the trained VAE latent space. This is important as we have used the latent space features from the reparametrization trick as input in our Logit model for the classification exercise.

First, we took two images from the test dataset. We encoded them to get their latent representations, interpolate between these representations, and decoded the interpolated latent vectors and generated 10 images.

We notice that the 10 images look very much in line with what we expect of the satellite images we have.

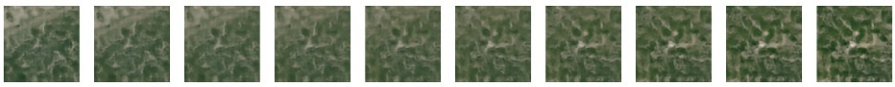

Figure 4: VAE Latent Space Interpolation

Second, we performed a random sampling of 10 images from the latent space. We noticed while visually inspecting these images that they are is in line with what is expected of the images in the dataset to be.

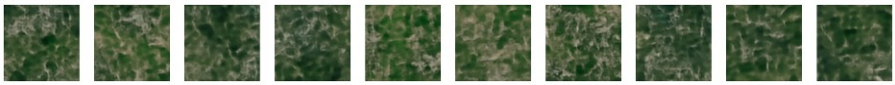

Figure 5: Random Sample From VAE Latent Space

Lastly, we visually inspected 10 reconstructed images: original images (top), reconstructed images (bottom). We noticed that the reconstructed images are very close to the original images.

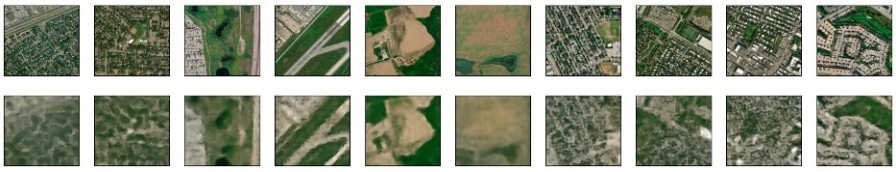

Figure 6: VAE Random Sample of Original Input Vs Reconstructed Input

3.3 Benchmark

For benchmark, we trained our dataset on ResNet-50 over 5 epochs. We obtained a confusion matrix very close to that of our VAE-Logit architecture. RestNet-50 generated an accuracy of 0.9978, close to only 2% higher than our VAE-Logit architecture, while producing an AUC and Precision quite aligned with our architecture.

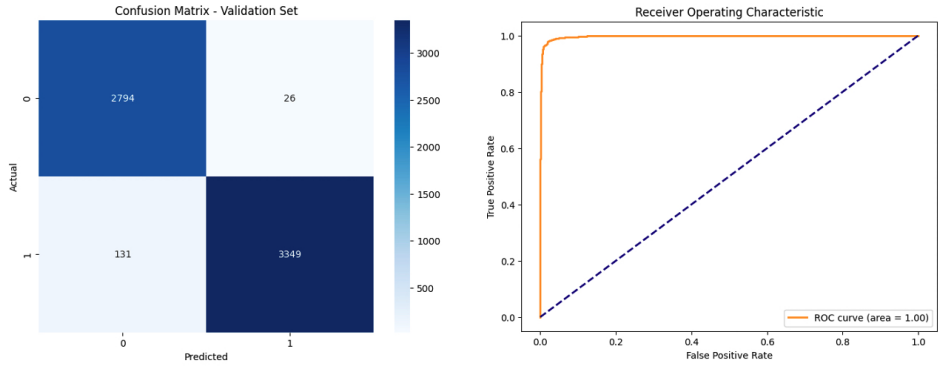

Figure 7: ResNet-50 Evaluation Confusion Matrix and ROC Curve

4 Analysis

Features extracted from the trained VAE model were visually inspected and showed that the model was actually retaining the most important features of the images while compressing the data to around 1% of its original dimension. This was validate while inspecting the reconstructed images and how close they were to the original images. These features from the latent VAE space along with the images geolocations (the images were labeled wildfire and no-wildfire) produced a Logit classifier that performed as good on the validation data as a RestNet-50 as shown the Figure 8 below.

Logit	RestNet-50
Accuracy: 0.9747	0.9978
AUC: 0.9978	0.9998
Precision: 0.97 and 0.98	0.96 and 0.99

Figure 8: Evaluation Logit Vs RestNet-50

Our VAE model took 0.4 hour to train an epoch whilte RestNet-50 took 3 hours to train one epoch on a MacBook Pro (November 2023 model) using CPU only. We noticed reduction in error between the first epoch trained and the last on both models. In total, the VAE model took 4 hours to train on 10 epochs. The Logit model took 40 minutes to train on the extracted features and images geolocations.

Overall, it took 5.5 hours to train, test, and validate our VAE-Logit architecture. On the other hand, ResNet-50 took 15 hours to train, test, and validate while generating similar performance.

5 Conclusion

In conclusion, the proposed VAE-Logit architecture for the classification of satellite image into wildfire and no-wildfire classes is a promising start to explore the monetizing of wildfire risk applied to value of property in a climate scenario analysis applied to mortgage portfolio. Our proposed architecture is not computational intensive as compared to other image processing deep learning established models such as ResNet-50 as seen abvove.