fire

MDPI

Article

Desert/Forest Fire Detection Using Machine/Deep Learning Techniques

Mason Davis [1] and Mohammad Shekaramiz [1,*]

Machine Learning and Drone Laboratory, Engineering Department, Utah Valley University, Orem, UT 84058, USA; mason.davis@uvu.edu
* Correspondence: mshekaramiz@uvu.edu

Abstract: As climate change and human activity increase the likelihood of devastating wildfires, the need for early fire detection methods is inevitable. Although, it has been shown that deep learning and artificial intelligence can offer a solution to this problem, there is still a lot of room for improvement. In this research, two new deep learning approaches to fire detection are developed and investigated utilizing pre-trained ResNet-50 and Xception for feature extraction with a detailed comparison against support vector machine (SVM), ResNet-50, Xception, and MobileViT architectures. Each architecture was tuned utilizing hyperparameter searches and trials to seek ideal combinations for performance. To address the under-representation of desert features in the current fire detection datasets, we have created a new dataset. This novel dataset, Utah Desert Fire, was created using controlled fires and aerial imaging with a DJI Mini 3 Pro drone. The proposed modified ResNet-50 architecture achieved the best performance on the Utah Desert Fire dataset, reaching 100% detection accuracy. To further compare the proposed methods, the popular forest fire detection dataset, DeepFire, was deployed with resulting performance analyzed against most recent literature. Here, our proposed modified Xception model outperformed latest publications attaining 99.221% accuracy. The performance of the proposed solutions show an increase in classification accuracy which can be leveraged for the identification of both desert and forest fires.

Keywords: desert fire; forest fire detection; deep learning; transfer learning; Xception; ResNet-50; support vector machines; image classification; hyperparameter tuning

Citation: Davis, M.; Shekaramiz, M. Desert/Forest Fire Detection Using Machine/Deep Learning Techniques. *Fire* **2023**, *6*, 418. https://doi.org/10.3390/fire6110418

Academic Editors: Washington Rocha, António Vieira and Marcos Francos

Received: 20 September 2023
Revised: 21 October 2023
Accepted: 23 October 2023
Published: 29 October 2023

1. Introduction

Wildfires are both ecological and economic disasters. These fires are characterized as uncontrolled and unpredictable fires in areas with combustible vegetation. According to recent reports, human-caused fires account for the majority at 89%, while other natural causes occur at a less frequent rate [1,2]. Natural causes such as lightning can be hard to detect as if they strike in remote areas, they may only be noticed after being developed into a sizeable fire. The destruction of environments due to fire also causes side effects on the ecosystem and surrounding communities. In desert environments, fire has been associated with the elimination of vegetation and increased probability of erosion [3]. These destructive fires also harm soil biological components that are vital for the maintenance of native plant species [3]. Catastrophic and fast-evolving fires also endanger and deteriorate the health of the surrounding communities. The smoke and particulates are known to cause lung and eye irritation while long exposure can lead to decreased lung function, exacerbation of asthma, and bronchitis [4].

Given deserts' inherently arid and barren nature, the susceptibility to wildfires is historically low. However, with the increase in invasive plant species and sporadic heavy rainfall, the flammability of these regions is increasing [5]. These implications are further highlighted in the recent Dome Fire located in the Mojave Desert, California. In 2020, a lightning strike caused a small fire that rapidly developed and destroyed 17,512 hectares (Ha)

of the national park, along with 1.3 million Joshua trees [6]. As another such instance, the York Fire has been California's largest wildfire of 2023 [7]. With the increased precipitation driving vegetation growth in the area, this fire destroyed 37,667 Ha of land [7]. It turns out that invasive species are an increasingly significant driver of desert fires. It has been shown that invasive plant species can quickly recover after fire, resulting in the cyclical increase of fire probability and the spread of invasive species [8,9]. While these factors contribute to increasingly violent desert fires, forest fires remain a major concern.

Forests not only provide resources such as food, fuel, and shelter to species but also help clean the air by absorbing carbon dioxide and preventing erosion by dissipating rainfall and slowing runoff with their root system [10]. While forest fires are not a new phenomenon, they are increasing in size and quantity inducing a larger loss of life, resources, and capital. Since 1960, the top five largest wildfires by acreage burned in the United States occurred from 2007–2020 [2]. Forest fires do not only affect the United States but also the global community at scale. Canada, which makes up 9% of the world's forests, is having one of its worst fire years to date. According to the National Wildland Fire Situation Report, as of 21 June 2023, there have been a reported 2765 fires which is already above the 10-year average of 2068. While the acreage burned is monumental, the area is just shy of 6 million hectares. This far exceeds the 10-year average of 393,746 hectares [11]. As another example, Russia is home to 20% of the world's forests and has seen a similar trend of devastating fire behavior. From 2001 to 2021, Russia lost 52.8 Mha of tree cover from fires with the greatest loss during the 2021 fire season [12].

The magnitude and quantity of fires have been increasing year over year and researchers have started to study the climate change trends in tandem with wildfire statistics. As a consequence of the greenhouse effect, the temperature is becoming higher globally with a predicted increase of 5–6 °C by the year 2100 [13]. Temperature extremes drive environments that foster volatile wildfires, increasing in quantity and intensity [14]. Brown et al. investigated the tie between climate change and wildfires [15]. By comparing the energy release component index from 1975 to 1996 against a model that calculated the expected rises due to climate change, they found that areas such as the Great Basin will undergo longer fire seasons [15].

Due to the large economic cost of fighting wildfires and the changing conditions that foster faster-growing fires, local and national leaders have pushed for preventative and control measures in recent years. Thanks to advances in computing power and the availability of state-of-the-art processing power, computer vision and deep learning have been recently applied to the problem of early fire detection. The recent boom in the popularity of neural networks and deep learning is due to the increase in computational performance and availability of modern graphics processing units (GPUs). GPUs can be used to implement these architectures and provide an excellent computational performance boost with up to 20× the speed of CPU-only processing [16]. This has been utilized by researchers for early fire detection to improve the training time and performance on large datasets. However, the desert fire detection using this technology has not been studied or addressed seriously.

Dawar et al. used a dataset consisting of satellite imagery of Canadian forest fires and it was shown that their proposed convolutional neural network (CNN) model outperformed Alexnet, Xception, MobileNet, and Lenet5 with the accuracy of 95.14% [17]. In another study, Nallakaruppan et al. compared deep learning architectures on satellite images of fire with DenseNet-201 outperforming Inception, ResNet-50, and VGG-16 with an accuracy of 98.46% [18].

By applying transfer learning and making use of pre-trained weights for the forest fire detection problem, researchers have seen increases in performance and accuracy. In this regard, Khan et al. proposed a transfer learning approach utilizing VGG-19 as the base network. When compared to K-nearest neighbors, SVM, Naïve Bayes, and Logistic regression, it was concluded that the transfer learning approach was superior with an accuracy of 95% on the DeepFire dataset [19]. Following their initial work, Khan and

Khan furthered their research and proposed a novel approach to fire detection refered to as FFireNet. FFireNet was developed using MobileNetV2 as the backbone and modifying the fully connected layers at the end with ReLU and sigmoid functions. This approach proved beneficial as it performed well on the DeepFire dataset achieving 98.42% accuracy and outperformed other architectures such as Xception, InceptionV3, and ResNet152V2 [20]. On the same dataset, a particle swarm-based federated learning approach was evaluated by Supriya and Gadekallu [21]. Overcoming the hurdles of communication lags and transmission processing power, their proposed approach achieved 94.47% accuracy on the test data [21]. Namburu et al. also leveraged transfer learning with MobileNet, adding a flattened, dense, and softmax layer for the purpose of fire classification. Their proposed method, X-MobileNet, achieved 98.89% accuracy on a large dataset created using drone footage [22]. Idroes et al. proposed TeutongNet with the use of ResNet-50 as their backbone architecture [23]. This architecture is made based on a pre-trained ResNet-50 and adding global average pooling, dense, dropout, and sigmoid layers. TeutongNet was trained and tested on the DeepFire dataset and achieved 98.68% accuracy [23]. Alice et al. also employed ResNet-50 as the feature extraction to their proposed fire detection model referred to as AFFD-ASODTL. By combining ResNet-50 with a Quasi-Recurrent neural network for classification, they were able to reach 97.33% accuracy and surpassed other architectures on their created dataset [24].

Another area of fire detection research that is increasing in popularity is real-time detection and classification. This area uses machine learning for classification as well as tracking and relaying data about the fire in real-time. In this case, Wu and Zhang investigated three architectures, You Only Look Once (YOLO), Single Shot Detector (SSD), and Region based convolutional neural networks (R-CNN), for real-time forest fire detection and their relative performance. The models were trained using images of smoke and fire in different environments, both forest and urban, to train the models to detect and segment both fire and smoke. Experimentally, they concluded that the YOLO algorithms had poor performance on small and cool fires. To resolve this issue, Wu and Zhang altered the structure of YOLO by adding one more convolutional and max-pooling layer. With these modifications, Faster R-CNN and SSD provided the best performance still at 99.7% and 99.88% for fire detection, respectively [25]. Jin and Lu made use of these real-time processing techniques to detect movement as part of a proposed fire detection process. This included the combination of real-time data, feature extraction, and classification for fire detection. Through comparative analysis of algorithms for extraction and classification steps, Jin and Lu concluded that AdaBoost and Naïve Bayes were the leading algorithms for low and high-dimensional classifiers, respectively. With these in place, the performance of their approach on evaluation data was 97.33% [26]. A large hurdle for real-time detection and edge computing is the balance of size and performance. To overcome the large computational costs of traditional real-time detection methods, in [27] the authors proposed a lightweight YOLOv7 architecture for deployment in UAV's for the purpose of smoke classification. Altering traditional YOLOv7 with reduced computational convolutions, activation functions, and the reassembly of features, their model provided high accuracy while significantly cutting size and computational cost. Measuring the Giga floating point operations per second (GFLOPs), this novel approach improved the performance by 6% over the baseline YOLOv7 [27].

While wildfires affect forest environments heavily, the review of existing literature revealed a need for the investigation of desert fire features. In order to investigate detection methods in these diverse environments, new datasets need to be created and are required to be fed to the models during the training period. This lack of desert feature representation in popular fire datasets was also highlighted in [28]. To tackle this problem, we propose a new dataset which includes 986 total fire and no-fire images for use with fire classification models. This new dataset, Utah Desert Fire, was created to analyze the models in fire feature extraction and classification abilities. This dataset was created in a desert environment which provides valuable insight into the capability of fire detection in diverse conditions.

In this paper, we explore the performance of four existing architectures and propose two novel approaches for fire classification. The models were tuned and trained on this newly proposed dataset with resulting performance comparisons drawn. Following these comparisons, the proposed models were trained and tested on the popular DeepFire dataset created by Khan et al. to compare performance with recent proposed architectures in the literature [19].

The following outlines the contributions and flow of this research:

- Review of the existing research in fire detection with the focus on desert and forest fires.
- Creation of a novel desert fire detection dataset, Utah Desert Fire.
- Proposal of modified transfer learning approaches for Xception and ResNet-50.
- Hyperparameter tuning and testing for the best performance on the proposed Utah Desert Fire dataset.
- Comparison of SVM, Xception, ResNet-50, and MobileViT architectures performance against proposed models on the Utah Desert Fire dataset.
- Comparison of our proposed models performance with exisiting solutions in the literature on the DeepFire dataset [19].

The following sections will discuss the architectures and approaches proposed, the methods used for tuning and finding best performance, followed by resulting performance and literature comparisons.

2. Materials and Methods

2.1. Architecture Overview

The four models/architectures investigated for early desert and forest fire detection include SVM, ResNet-50, Xception, and MobileViT. Modified architectures making use of transfer learning were created as well using ResNet-50 and Xception as the backbone. The following sections discuss the architectures in more detail along with the proposed transfer learning approaches investigated in this research.

2.2. SVM

SVM is a popular classification and regression algorithm that transforms data into a higher dimension to separate the data with hyperplanes. Once the data is separated with the initial hyperplane, support vectors are created by passing two more hyperplanes through one or more data points on each side. In our case, fire and no-fire classes create the two hyperplanes that separate our data. Optimization involves creating the largest distance, or maximum margin, between the support vectors for optimal performance. The regularization parameter C, also known as the penalty parameter, is used to determine the strength of regularization. This adjusts the amount of bias for misclassification vs. margin size in the algorithm. An important aspect of SVM is the use of kernals which decide how the data is manipulated for separation and provides the shape of hyperplanes [29]. Figure 1 illustrates a simple hyperplane separation between two classes. The support vectors can be seen passing through the nearest data points to the middle-most vector. Noting this illustration is a simple linearly separable example of an SVM.

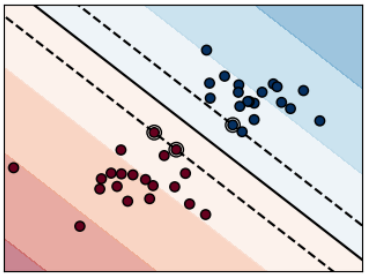


Figure 1. An SVM example. Here, the filled circles show the data points, the solid line represents the decision boundary, and the dashed lines represent the support vectors [30].

2.3. ResNet-50

The ResNet family of convolutional neural networks was proposed by He et al. [31]. As neural networks increase in depth, they become more challenging to train due to backpropagation loss. To remedy this, He et al. proposed skip connections that create a shortcut for backpropagation to reach earlier layers and eliminate the vanishing gradient problem [31]. Residual connections are also utilized, giving to the name ResNet, which learn residual functions regarding the input layer. ResNet can be created using various layer counts, i.e., ResNet-18, ResNet-34, ResNet-50, etc. In this paper, ResNet-50 is employed for fire detection simulations. ResNet-50 was chosen by considering its size and performance. For use with smaller datasets, a 50 layer architecture will be sufficient for classification over 101 or 152 layer variants while also providing a significant calculation reduction. Figure 2 illustrates the ResNet-50 architecture with its notable skip connections between each stage. Each stage is repeated, indicated in the figure by the numbers below, dependent on the layer count desired. The convolutional filter size is also determined by the resulting size desired and can be seen in Figure 2.

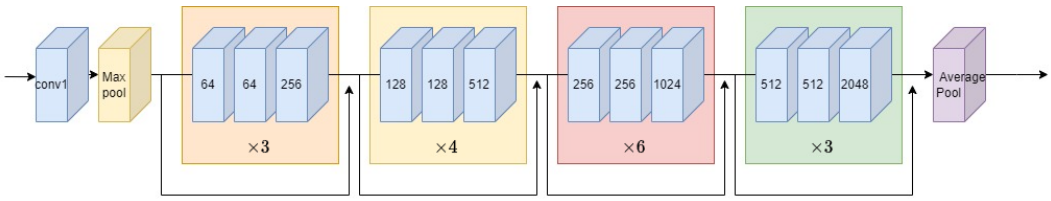


Figure 2. ResNet-50 architecture.

2.4. Xception

Xception is a deep convolutional neural network architecture created by Chollet to outperform the existing architectures in computer vision tasks [32]. Built based on InceptionV3, Xception differentiates itself by utilizng depthwise separable convolutions instead of traditional convolutions. This consists of a single convolutional filter for each input channel and a final pointwise convolution applied to all channels. This approach, known as depthwise separable convolutions, reduces the computations needed which make Xception faster and more efficient than traditional CNNs. Xception incorporates these convolutions into a 71-layer deep architecture that outperformed InceptionV3 on ImageNet and other large-scale datasets [32]. The Xception architecture is depicted in Figure 3 with three distinct sections, entry, middle, and exit. The entry and exit flow occur only once while the middle flow repeats 8 times before feeding into the exit flow. Another key feature is the skip connections utilized throughout all flows although, only the entry flow makes use of a convolutional layer between each skip connection.

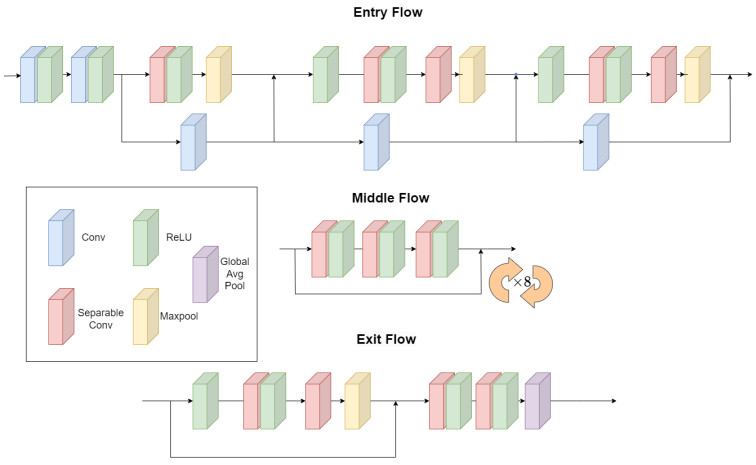


Figure 3. Xception architecture.

2.5. MobileViT

MobileViT was created in 2022 by researchers at Apple with mobile computing performance in mind [33]. This model, as illustrated in Figure 4, combines the strength of convolutions and transformers into one light weight model. Traditionally, transformers are heavy weighted networks that were developed for natural language processing tasks, but have since been adapted to computer vision in recent years. Vision transformers (ViTs) use self-attention mechanisms to learn long-range dependencies and relationships between different pixels in images. This has been shown to be a competitive approach to image classification tasks while remaining cheap to train [34]. By using vision transformers in tandem with convolutions, the model is able to learn from both the global and spatial information of the image. MobileViT is a light weight, general purpose vision transformer containing 1.3M parameters in the xxs variant used in this research [33]. Due to the potential for on device execution, MobileViT will serve as a great comparison against the larger models in Xception and ResNet.

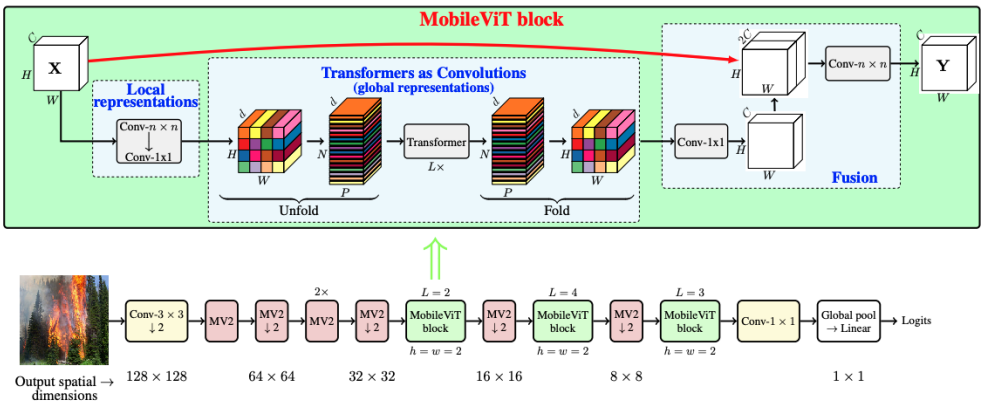


Figure 4. MobileViT model [33].

2.6. Transfer Learning

Transfer learning is the concept of networks not starting from scratch. This is accomplished by taking a models knowledge previously learned on a task and leveraging those weights for something new. In computer vision, models are trained on large datasets such as ImageNet, an extensive visual database with more than 14 million images [35]. These models learn the key features of images and allow for use in other classification tasks via transfer learning. This approach consists of loading those learned weights and freezing each layer to create the feature extraction layer. A new predictive layer is then added to allow for a model to fine-tune on a new dataset for a specific problem. Thus, a model can be trained faster, with less data, and with higher performance. This approach seems promising especially when datasets are small and resources are limited, allowing learned generalization on large datasets to accelerate new classification problems [36].

2.7. Proposed Modified Xception

The modified Xception model utilizing transfer learning is built on an Xception model with pre-trained weights on ImageNet which are frozen during training [35]. This allows leverage of the pre-trained model's feature extraction for fire detection. Connected to the Xception base model, a rectified linear unit activation (ReLU), average pooling layer, and sigmoid activation are added. These form the new layers of the model and allow for binary classification, in our case fire and no-fire. ReLU introduces non-linearity to our model which helps prevent the vanishing gradients. This layer is then followed by average pooling to down sample our feature map by averaging the values defined in the filter. This reduction in features prevents over fitting while also making our model computationally efficient. This is followed by the sigmoid activation that provides our classification of 1 or 0, fire or no-fire. Table 1 displays the layers of the proposed modified Xception as

discussed along with their output shapes and trainable parameters. The non-trainable parameters seen here indicate the frozen layers within the feature extraction base while the trainable parameters come from our newly added layers that are fine tuned for fire classification. Figure 5 shows the built architecture of the proposed modified Xception model that visualizes the information seen in Table 1.

Table 1. Proposed Modified Xception Summary.

Layer (Type)	Output Shape	Parameters
Input Layer	(250, 250, 3)	0
Xception	(8, 8, 2048)	20,861,480
ReLU	(8, 8, 2048)	0
Global Average Pooling	(None, 2048)	0
Sigmoid	(None, 1)	2049
Total Params: 20,863,529		
Trainable Params: 2049		
Non-trainable params: 20,861,480		

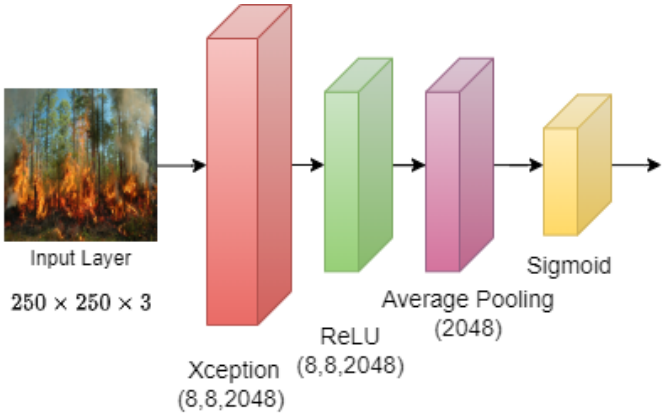


Figure 5. Proposed modified Xception model.

2.8. Proposed Modified ResNet-50

Utilizing the same methodology as the proposed modified Xception model, a modified ResNet-50 was built leveraging transfer learning. This allows for weights learned on ImageNet to be loaded and frozen making use of the achieved feature extraction capabilities for our fire detection problem. Adding new layers: ReLU, max pooling, and sigmoid to form our predictive layer. Figure 6 shows the entire architecture for the proposed modified ResNet-50. The input layer is a 250 × 250 × 3 image which is fed into ResNet-50 with weights learned from ImageNet training. This is then passed through a ReLU to add non-linearity and a max pooling layer to downsize our feature map. Finally, the sigmoid layer allows for binary classification and provides 2049 trainable parameters to fine-tune. Table 2 shows the parameters for each of these layers, noting that the only trainable layers are from our sigmoid output layer.

Table 2. Proposed Modified ResNet-50 Summary.

Layer (Type)	Output Shape	Parameters
Input Layer	(250, 250, 3)	0
ResNet-50	(8, 8, 2048)	23,587,712
ReLU	(8, 8, 2048)	0
Global Max Pooling	(None, 2048)	0
Sigmoid	(None, 1)	2049
Total Params: 23,589,761		
Trainable Params: 2049		
Non-trainable params: 23,587,712		

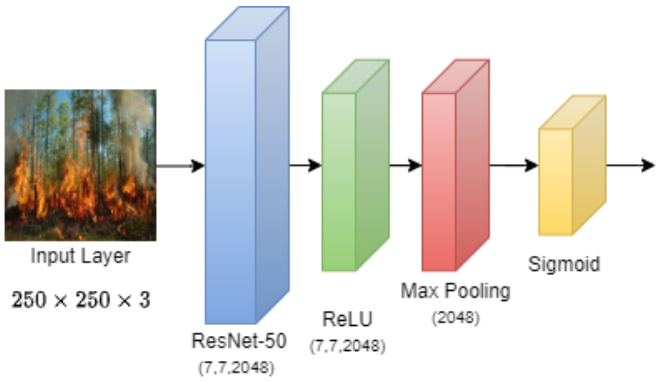


Figure 6. Proposed modified ResNet-50 model.

3. Simulations and Results

In this section we discuss the simulation settings and data used for training and evaluating our deep learning models on fire detection. The architectures discussed in Section 2 are fine-tuned and compared through multiple simulations for optimal performance. Performance metrics are outlined through which we make use of in the resulting comparisons of performance on the proposed Utah Desert Fire dataset. Finally, the proposed modified architectures are compared against recent literature on the popular fire detection dataset, Deepfire [19].

3.1. Simulation Environment

All simulations were performed using TensorFlow 2.10, Keras 2.10, and Python 3.9 on the following hardware: Intel® Core (Santa Clara, CA, USA) i5-7600K, 16 GB DDR4, NVIDIA GeForce GTX 1070 with 8 GB GDDR5.

3.2. Metrics

The following metrics are used for comparing the relative performance between our models and those proposed in recent literature. A positive label refers to the fire class or an image that depicts fire. Negative refers to the non-fire class or an image that does not depict fire. As predictions are made using the model, if the model correctly predicts the label, it will be defined as a true negative (TN) or true positive (TP) respective to the class in question. False positive (FP) is defined as predicted positive while its true class is negative. Similarly, false negative (FN) is defined as predicted negative with its true class being positive. These four definitions are used to create a visual representation of a model's prediction known as a confusion matrix. Along with this, the following performance metrics were calculated:

Accuracy is defined as the percentage of correct guesses a model makes, which is the correct guesses divided by all guesses made.

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN}) \quad (1)$$

Precision measures how well a model predicts the positive label, in this case fire, providing a measure for fire detection.

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP}) \quad (2)$$

Recall measures how completely the label in question is predicted. Note that the key difference between precision and recall is that recall considers FN predictions.

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN}) \quad (3)$$

F1-Score is the harmonic mean of precision and recall, combining the quality and completeness into one score.

$$\text{F1-Score} = (2 \times \text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (4)$$

3.3. Dataset Overview

With research emerging in the area of forest fires, there was a lack of desert environment representation in the publicly available datasets. For this reason, we have created a new dataset, Utah Desert Fire, to train and compare models on the important task of fire detection in this environment. To create the dataset, experiments were conducted which included the use of a Lion BullEx Intelligent training system, a DJI Mini 3 Pro (Figure 7), and a small controlled brush fire to capture fire and no-fire images. Figure 8 illustrates the experimental setup for the two conducted experiments which allowed for fire and no-fire images to be taken. Samples of the Utah Desert Fire dataset are illustrated in Figure 9.

Figure 7. DJI Mini 3 Pro Drone.

(**a**) Experiment setup for BullEx fire simulations. (**b**) Experiment setup for controlled brush fire.

Figure 8. Utah Desert Fire dataset experiments.

The dataset includes a total of 986 RGB images of size 250 × 250 which are further split for use in training and testing of our proposed models. Table 3 outlines the split totals which consists of 80% of the total images for training and 20% for testing, while 20% of the training data was further split for validation.

Table 3. Utah Desert Fire training split.

Label	Train	Validation	Test	Total
Fire	316	79	99	493
No-Fire	315	78	99	493
Total	631	157	198	986

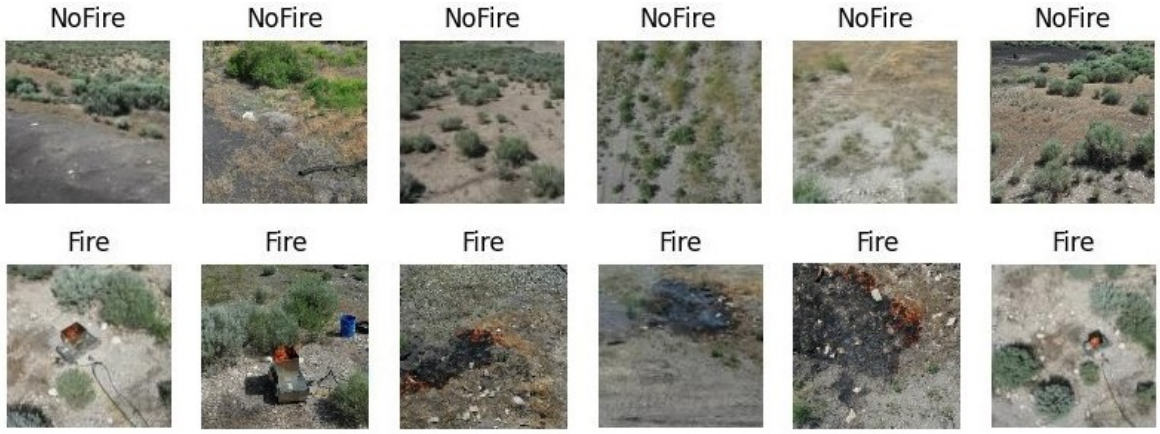


Figure 9. Sample images from the new fire dataset i.e., Utah Desert Fire dataset. Size of 250 × 250.

All models utilized pre-processing layers to normalize pixels from 0–255 to 0–1 for training. However, only the proposed modified transfer learning architectures took advantage of image augmentation. The list of augmentations used for the proposed modified ResNet and Xception are outlined in Table 4. Each deep learning architecture was trained over 50 epochs with binary cross-entropy as the loss function and input size of 250 × 250 as illustrated in Table 5. The following sections outline the hyperparameter tuning and simulations that were conducted to locate the best performance of each architecture on the Utah Desert Fire dataset.

Table 4. The augmentation applied to proposed modified models.

Random Augmentation Type	Range
Rotation	0–50°
zoom	±0.1%
Shear	0.1%
Translation	0.1–0.2%

Table 5. Simulation settings and parameters.

Simulation Parameter	Value
Epochs	50
Loss Function	binary cross-entropy
Image Size	250 × 250
Batch Size	32 *

* The proposed modified Xception and modified ResNet-50 utilized batch sizes of 64.

3.4. Support Vector Machine Results

Implementing an SVM for fire detection and classification was accomplished through the Scikit-Learn library [30]. The hyperparameters and functions of interest are the C_value, kernel function, and the degree of polynomial kernels. All other values were kept at their default values. Four C_values were tested on each of the five kernels. The models were then evaluated on the test data and the top results for each C_value are reported in Table 6.

Table 6. SVM results on the test data.

C_Value	Kernel	Accuracy (Test Data)	Precision	Recall	F1-Score
100	**RBF**	**90.909%**	**0.909**	**0.909**	**0.909**
10	**RBF**	**90.909%**	**0.909**	**0.909**	**0.909**
1	RBF	90.404%	0.905	0.904	0.904
0.1	Poly-degree 3	83.838%	0.838	0.838	0.838

According to Table 6, the radial basis function (RBF) was the preferred kernel through our hyperparameter search. RBF resulted in the highest accuracy of 90.909% for both C_values of 10 and 100. Polynomial of degree 3 was preferred when the C_value was the smallest at 0.1. Note that the larger the C_value, the smaller the margin becomes which provides a greater bias towards correct classification which is reflected in the resulting performance.

3.5. ResNet-50 Results

Here, ResNet-50 was simulated to determine the best performance. In order to find the best optimizer for each architecture, a random search of optimization algorithms was conducted utilizing KerasTuner [37]. The following optimizers were chosen: Accelerated Adaptive Moment Estimation (Adam), Nesterov-Accelerated Adaptive Moment Estimation (Nadam), Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSprop), and Adaptive Gradient Algorithm (Adagrad). These optimizers were ran at their corresponding default parameters initially as detailed in Table 7. The results in Table 8 outline the top 3 performing optimizations found through hypersearching.

Table 7. Optimization algorithms with default hyperparameters.

Optimization	Learning Rate	β_1	β_2	Momentum	ε	Initial Accumulator Value	ρ
Adam	0.001	0.9	0.999	-	1×10^{-7}	-	-
Nadam	0.001	0.9	0.999	-	1×10^{-7}	-	-
Adagrad	0.001	-	-	-	1×10^{-7}	0.1	-
RMSprop	0.001	-	-	0.0	1×10^{-7}	0.1	0.9
SGD	0.01	-	-	0.0	-	-	-

Table 8. ResNet-50 initial optimizer search top three performance.

Optimizer	Accuracy (%)	Precision	Recall	F-Score
Adam	99.495%	1.0	0.990	0.995
SGD	99.490 %	1.0	0.990	0.995
Nadam	98.990%	1.0	0.980	0.990

As seen in Table 8, the Adam and SGD optimizers provided the best performance followed by Nadam. The highest accuracy achieved from the initial optimizer search was 99.495% with the Adam optimizer. To seek further performance gains from the model, these top 3 optimizers were tuned. This was also achieved through KerasTuner [37].

The search space for the following simulation included learning rate, momentum, β_1, β_2, and ρ. The learning rate parameter provides how much the optimization adjusts the weights through each pass as it minimizes error. Momentum is a parameter of gradient descent in which the update term gets an added momentum value to assist optimization over local minima and speeds up the convergence. β_1 and β_2 are parameters used in Adam and Nadam optimizers. β_1 controls the weight of the first moment estimates while β_2 controls the weight of the second moment estimates. Finally, ρ adjusts the decay factor which helps prevent an optimizer from overshooting minimum loss. These values were randomly selected and trained on the ResNet-50 architecture over 50 epochs with validation loss being monitored to determine the best results. The best model for each algorithm was then tested on the test data for comparison against the first default simulation as shown in Table 8.

Table 9 details the performance of ResNet-50 after experimentally searching for optimal hyperparameters. Here, it is shown that SGD has the highest accuracy reaching 99.495% with the optimal found tune. When compared to the default simulation in Table 8, the performance was not increased through further tuning.

Table 9. ResNet-50 top three optimization algorithms with the best performing hyperparameters.

Optimization Algorithm	Learning Rate	Momentum	β_1	β_2	Nesterov Momentum	Test Accuracy
SGD	**0.01**	**0**	**-**	**-**	**False**	**99.495%**
Adam	0.001	-	0.01	0.001	-	98.485%
Nadam	0.01	-	0.001	0.00001	True	98.485%

3.6. Xception Results

The same methodology was applied to determine the best optimization algorithm for the Xception architecture. The default hyperparameters were utilized with the Keras API [38] and simulated over 50 epochs, while keeping the best result. Table 10 presents the top 3 optimizers and their corresponding test accuracy from the initial search which includes the 5 optimizers discussed in Table 7. Here, it is shown that Adam, RMSprop, and SGD provide the highest performance for the Xception architecture. RMSprop provided the highest accuracy at 99.495% followed by Adam and SGD at 99.490% and 98.485%, respectively.

Table 10. Xception initial optimizer search top three performance.

Optimizer	Accuracy (%)	Precision	Recall	F-Score
Adam	99.490%	1.0	0.990	0.995
RMSprop	99.495%	0.990	1.0	0.995
SGD	98.485%	1.0	0.970	0.985

Once the best optimization algorithms were determined, the hyperparameters were further searched for an optimal performance. This was accomplished using KerasTuner [37] which allowed quick and seamless hyperparameter search. The top three algorithms were used in tandem with 20 random simulations, including random variations for learning rate, β_1, β_2, ρ, and Momentum (SGD). The top results are tabulated in Table 11.

Table 11. Xception top three optimization algorithms with the best performing hyperparameters.

Optimization Algorithm	Learning Rate	Momentum	ρ	β_1	β_2	Nesterov Momentum	Test Accuracy
RMSprop	**0.0001**	**0.9**	**0**	-	-	-	**100%**
Adam	0.01	-	-	0.0001	0.00001	-	99.495%
SGD	0.01	0.5	-	-	-	True	99.495%

Through hyperparameter search, the RMSprop optimization algorithm achieved a 100% accuracy score on the test data. The hyperparameters that provided the best accuracy were a learning rate of 0.0001, momentum of 0.9, and $\rho = 0$. This tuned simulation achieved a 0.5% accuracy increase over the default RMSprop hyperparameters. Both the Adam and the SGD optimizers realized marginal accuracy gains of ≈0.005% when compared to the initial optimizer search in Table 10.

3.7. MobileViT Results

Following the same tuning method, MobileViT was simulated to find the optimal performing hyperparameters. As seen in Table 12, the best performing optimizers were

found to be Adam, RMSprop, and SGD. Here, it can be seen that the performance was similar between the top three. Although, Adam attained the best accuracy of 99.696%.

Table 12. MobileViT initial optimizer search top three performance.

Optimizer	Accuracy (%)	Precision	Recall	F-Score
Adam	99.696%	0.996	0.998	0.997
RMSprop	99.492%	0.990	1.0	0.995
SGD	99.391%	0.994	0.996	0.994

Seeking further performance increases with the MobileViT architecture, these top three optimizers were searched for top performing hyperparameter tunes. The search results are shown in Table 13 with corresponding parameter selection. Here, it can be seen that the Adam optimizer continues to achieve highest detection accuracy with 99.797%. This tune realized a gain of 0.1% accuracy over the default hyperparameters seen in Table 12. SGD also achieved a higher accuracy over the default values by a similar margin while RMSprop dropped slightly. However, the best performing hyperparameters for RMSprop as seen in Table 13 are the default values. Thus, these parameters are consistent in providing optimal performance.

Table 13. MobileViT top three optimization algorithms with best performing hyperparameters.

Optimization Algorithm	Learning Rate	Momentum	ρ	β_1	β_2	Test Accuracy
Adam	**0.001**	-	-	**0.001**	**0.1**	**99.797%**
SGD	0.001	0.99	-	-	-	99.495%
RMSprop	0.001	0	0.9	-	-	99.290%

MobileViT is the smallest model in our comparison, with the selected xxs variant containing 1.306 million (M) parameters, but performs competitively against the larger models in accuracy. Section 3.11 draws a full comparison of all tuned model performance where the model size vs. accuracy is clearly illustrated and further discussed.

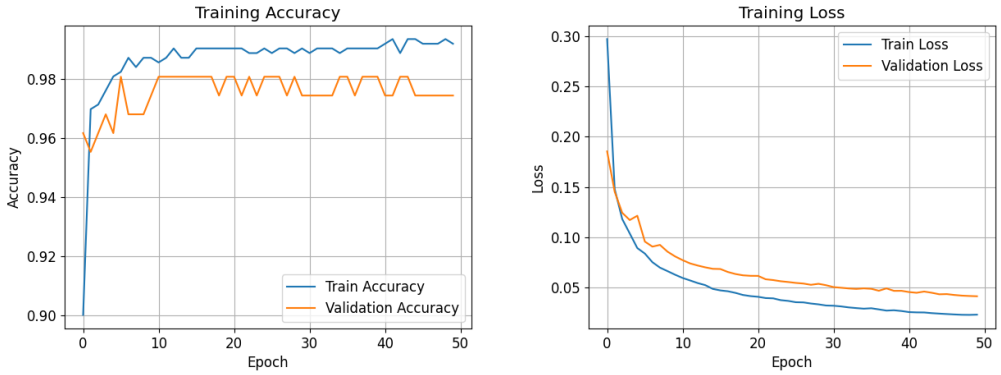
3.8. Proposed Modified ResNet-50 Results

A modified ResNet-50 was created using pre-trained weights trained on ImageNet as the feature extraction base [35]. Additional layers were added including a ReLU non-linearity, max pooling, and an output layer with sigmoid activation to form the classification layers. Testing included max pooling and average pooling where it was found that max-pooling achieved the best results for this model. Here, batch sizes of 32 and 64 were investigated, with 64 providing better results overall. Data augmentation was also used through all training as was described in Table 4. Leveraging the optimization and optimization parameter searches conducted on base ResNet-50, the optimizer chosen for this model was SGD, as it provided the best accuracy and generalization. The model was trained over 50 epochs with input size of 250 × 250. Detailed description of the training parameters are tabulated in Table 14.

Table 14. Proposed modified models training parameters.

Setting	Modified ResNet	Modified Xception
Batch Size	64	64
Epochs	50	50
Optimizer	SGD	Adam
Learning Rate	0.01	0.001

Training accuracy and loss performance over the 50 epochs is illustrated in Figure 10. Here, it can be seen the Proposed modified ResNet-50 deploying transfer learning quickly reaches 98% accuracy in the validation set with the loss continuing its downwards trend well past epoch 40. This model achieved 100% accuracy on the Utah Desert Fire test set, showcased in the confusion matrix Figure 11. The test performance receiver operating curve (ROC) and precision-recall (PR) curve for this proposed method are illustrated in Figure 12.



(**a**) (**b**)

Figure 10. Proposed modified ResNet-50 training and loss performance on Utah Desert Fire. (**a**) Proposed modified ResNet-50 training accuracy. (**b**) Proposed modified ResNet-50 training loss.

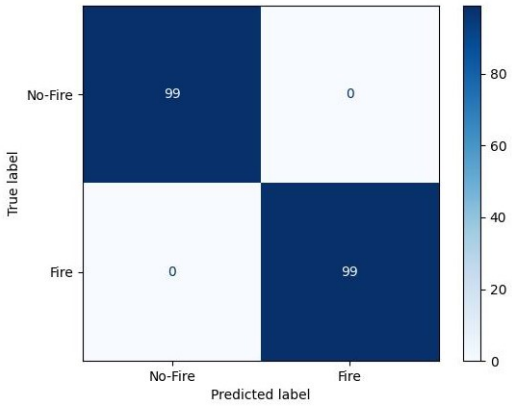
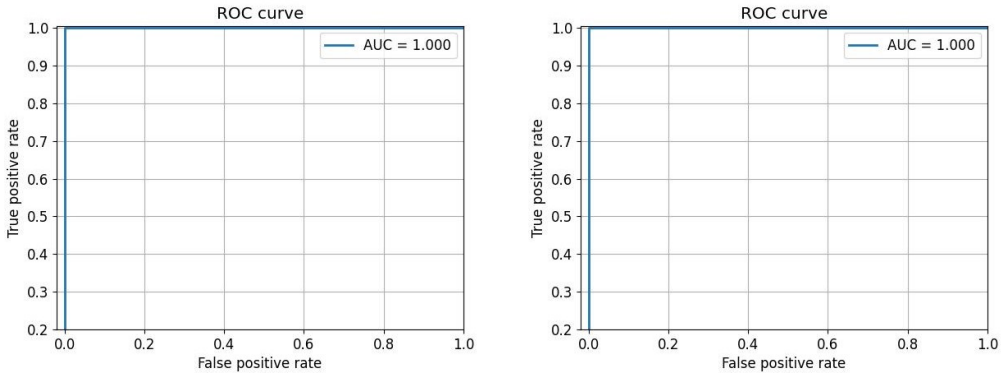


Figure 11. Proposed Modified ResNet-50 Utah Desert Fire test classification.



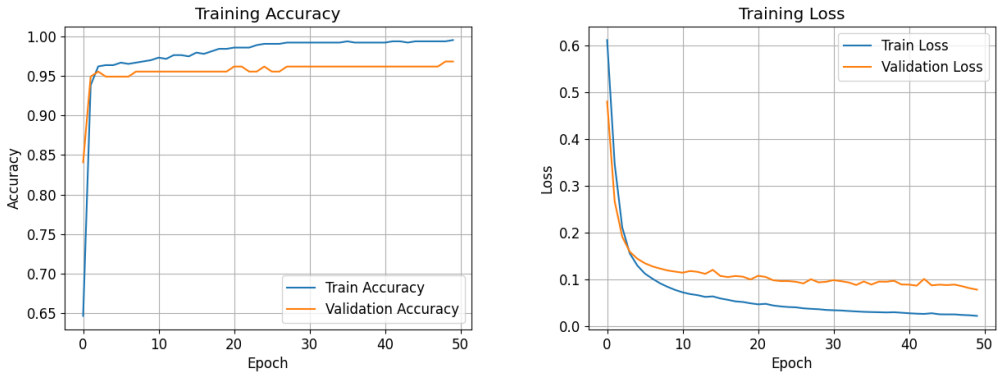
(**a**) (**b**)

Figure 12. Proposed modified ResNet-50 test performance on Utah Desert Fire. (**a**) Proposed modified ResNet-50 ROC curve. (**b**) Proposed modified ResNet-50 PR curve.

3.9. Proposed Modified Xception Results

Similar to the proposed modified ResNet-50, a modified Xception architecture was also created, simulated, and examined. Here a pre-trained Xception architecture was used as the feature extraction layer. This was achieved by applying the weights learned from training on the ImageNet dataset and freezing those layers for subsequent training [35]. The feature extraction layer was fed into a ReLU non-linearity, average pooling, and sigmoid which formed our prediction layers for fire detection. These additional layers were fine-tuned to our new task of fire classification. Both max and average pooling were tested, with average pooling consistently providing better performance on the Utah Desert Fire test data. Based on the results obtained from the optimizer search on base Xception, the RMSprop optimizer was chosen for our modified Xception simulations. Batch size was altered from the original Xception simulations above to 64 which provided consistent performance increases during training and testing.

Figure 13 shows the training accuracy and loss over the 50 epoch training period for the proposed modified xception architecture. The model learned rapidly over the first 5 epochs, reaching 95% accuracy on the validation set. As seen in Figure 14, the model only classified one image incorrectly in the test set resulting in 99.495% accuracy. The ROC and PR curves for this simulation can be seen in Figure 15.



(**a**) (**b**)

Figure 13. Proposed modified Xception training and loss performance on Utah Desert Fire. (**a**) Proposed modified Xception training accuracy. (**b**) Proposed modified Xception training loss.

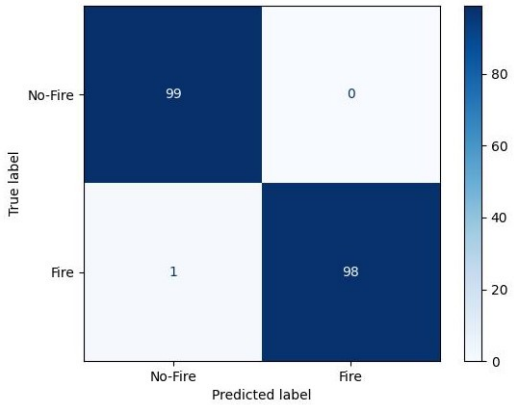
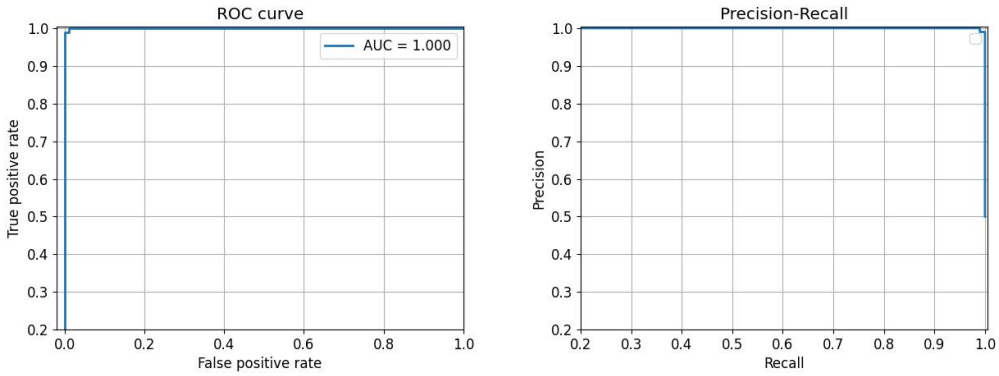


Figure 14. Proposed Modified Xception Utah Desert Fire test classification.



(**a**) (**b**)

Figure 15. Proposed modified Xception test performance on Utah Desert Fire. (**a**) Proposed modified Xception ROC curve. (**b**) Proposed modified Xception PR curve.

3.10. Architecture Comparison

Table 15 draws a full comparison of the models investigated in this paper and their performance on the proposed Utah Desert Fire dataset. Here, it is shown that both the proposed modified ResNet-50 model and base Xception exceed in accuracy reaching 100%. This was followed by MobileViT at 99.797% accuracy and the proposed modified Xception model and base ResNet-50 at 99.495%. Lastly, the performance of SVM trails these newer deep learning models however, still achieves 90.909% accuracy and provides a baseline comparison as a simple algorithm. Table 15 also draws comparison to each perspective models parameter size in millions. The inherent trade off between large heavy weight models and small light weight models is not clearly seen in this dataset. MobileViT, being the smallest sized model in our comparison is able to classify as well as the other heavy weight models in Xception and ResNet-50. For cases of edge deployment and real-time image processing the trade off is vital for achieving real time inference, as large bulky models are both computationally expensive and occupy more memory.

Table 15. Performance on Utah Desert Fire.

Model	Accuracy	Precision	Recall	F1-Score	Model Size (Params)
Proposed Modified ResNet-50	**100%**	**1.0**	**1.0**	**1.0**	**23.590 M**
Xception	**100%**	**1.0**	**1.0**	**1.0**	**20.861 M**
MobileViT	99.797%	1.0	0.996	0.998	1.306 M
Proposed Modified Xception	99.495%	1.0	0.990	0.995	20.864 M
ResNet-50	99.495%	1.0	0.990	0.995	23.587 M
SVM	90.909%	0.909	0.909	0.909	-

3.11. Comparison with Recent Literature on the DeepFire Dataset [19]

With the surge in popularity, machine learning and artificial intelligence has been the subject of research for fire detection in recent years. One popular forest fire dataset used in this area was created by Khan et al. and is known as the DeepFire dataset [19]. This section draws a comparison between the present literature and corresponding performance with our proposed architectures. Table 16 illustrates the latest performance on the DeepFire dataset. According to Table 16, the proposed modified Xception architecture utilizing transfer learning has achieved the highest accuracy at 99.211%. Idores et al.'s TeutongNet achieved the second-highest accuracy on this dataset, surpassing Khan and Khan's FFireNet. Both methods utilized a transfer learning approach similar to our proposed modified Xception but, made use of differing pre-trained base models and classification layers [20,23]. The top three performing architectures indicate how powerful transfer learning can be for image classification with small datasets.

Table 16. Comparison of DeepFire classification models.

Method	Accuracy	Precision	Recall	F1-Score
Proposed Modified Xception	**99.211%**	**1.000**	**0.9842**	**0.9920**
Idroes et al., 2023 [23]	98.680 %	0.9947	0.9793	0.9868
Khan & Khan, 2022 [20]	98.421%	0.9742	0.9947	0.9844
Proposed Modified ResNet-50	98.421%	0.9792	0.9895	0.9844
Mousavi & Ilanloo, 2023 [39]	96.880%	0.9736	0.969	0.9861
Khan et al., 2022 [19]	95.000%	0.9572	0.9421	0.9496
Supriya & Gadekallu, 2023 [21]	94.470%	-	-	-

4. Conclusions

Wildfires are devastating catastrophes that affect communities locally and worldwide. With the increasing temperature, dry conditions due to climate change, and the quick spread of such fires, it is vital that these disasters be detected early. This research outlined four popular machine/deep learning architectures and algorithms along with two new proposed modified models with their corresponding performance on the fire detection problem. Utilizing the newly created Utah Desert Fire dataset, the models were tuned for their optimal performing hyperparameters and compared to analyze performance. The highest performing model on the Utah Desert Fire dataset was found to be the proposed modified ResNet-50 model and base Xception with both reaching 100% accuracy after training, followed by MobileViT and our proposed modified Xception with 99.797% and 99.495% accuracy, respectively. The important tradeoff between model size and accuracy was compared and the smallest model, MobileViT, still achieved notable accuracy. For further comparison, the proposed modified models making use of transfer learning were tuned and trained on the popular forest fire dataset DeepFire. Here, our proposed modified Xception model achieved highest accuracy at 99.211% topping the accuracy of recent literature. These results show promising performance in accurately identifying both desert and forest fires which is a crucial step towards easing the environmental impact of these disasters.

5. Future Work and Challenges

Wildfires are fast spreading environmental disasters that require massive resources to curb their spread once started. We have seen the power of deep learning in identifying fires in both desert and forest settings. However, accurate detection is only part of the solution. Due to the rapid changes and spread of fires, detection speed also plays a vital role in wildfire prevention. Achieving real-time performance will require researchers to address the following challenges.

- Reduction in model complexity for faster inference speed.
- Robust datasets for generalization within diverse environments.
- Real-time location systems for relaying size and direction of fires to corresponding officials.
- Real-time data acquisition, processing, and transmission in remote environments.

Our future work will include the investigation of light weight models and methods for increasing the inference speed to achieve real-time performance, while having high detection accuracy.

Author Contributions: Conceptualization, M.S.; methodology, M.S. and M.D.; software, M.D.; validation, M.S. and M.D.; formal analysis, M.D. and M.S.; investigation, M.S. and M.D.; resources, M.D. and M.S.; data curation, M.D.; writing—original draft preparation, M.D.; writing—review and editing, M.S. and M.D.; visualization, M.D.; supervision, M.S.; project administration, M.S.; funding acquisition, M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This project was supported by an internal GEL-Quick grant from Utah Valley University.

Data Availability Statement: The created Utah Desert Fire dataset can be accessed at https://www.kaggle.com/datasets/mdavis9/utah-desert-fire-dataset (accessed on 10 August 2023). The proposed modified xception and ResNet architectures and code can be accessed at https://github.com/Noodler-Doodler/FireProject (accessed on 26 August 2023). The DeepFire dataset can be accessed at https://www.kaggle.com/datasets/alik05/forest-fire-dataset [19] (accessed on 5 May 2023).

Acknowledgments: The authors are also incredibly grateful for the full support of Justin Sprague, UVU Fire Marshall, for the controlled fire experiments in an open field that enabled us to take aerial images from the designated fire.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Adam	Accelerated Adaptive Moment Estimation
Adagrad	Adaptive Gradient Algorithm
FN	False Negative
FP	False positive
GFLOPs	Giga Floating Point Operations per Second
GPU	Graphics Processing Unit
Ha	Hectares
M	Million
Nadam	Nesterov-Accelerated Adaptive Moment Estimation
PR	Precision-Recall
RBF	Radial Basis Function
R-CNN	Region Based Convolutional Neural Networks
ResNet	Residual Network
RMSprop	Root Mean Square Propagation
ROC	Receiver Operating Characteristic
SSD	Single Shot Detector
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
VGG	Visual Geometry Group
ViTs	Vision Transformers
Xception	Extreme Inception
xxs	Extra Extra Small

References

1. Wildfire Statistics—CRS Reports. Available online: https://crsreports.congress.gov/product/pdf/IF/IF10244 (accessed on 23 June 2023).
2. National Interagency Coordination Center Wildland Fire Summary and Statistics Annual Report 2022. Available online: https://www.nifc.gov/nicc/predictive-services/intelligence (accessed on 23 June 2023).
3. Lynn, F. *Fire Impacts on the Mojave Desert Ecosystem: Literature Review*; U.S. Department of Energy Office of Scientific and Technical Information: Washington, DC, USA, 2012.
4. Wildfires, World Health Organization. Available online: https://www.who.int/health-topics/wildfires (accessed on 23 June 2023).
5. McLauchlan, K.K.; Higuera, P.E.; Miesel, J.; Rogers, B.M.; Schweitzer, J.; Shuman, J.K.; Tepley, A.J.; Varner, J.M.; Veblen, T.T.; Adalsteinsson, S.A.; et al. Fire as a fundamental ecological process: Research advances and frontiers. *J. Ecol.* **2020**, *108*, 2047–2069. [CrossRef]
6. U.S. National Park Service. Available online: https://www.nps.gov/moja/learn/nature/dome-fire.htm (accessed on 19 October 2023).
7. York Wildfire Still Blazing, Threatening Joshua Trees in Mojave Desert, Claire Thornton, USA Today. Available online: https://www.usatoday.com/story/news/nation/2023/08/02/california-york-fire-update/70511816007/ (accessed on 19 October 2023).
8. Allen, E.B.; Steers, R.J.; Dickens, S.J. Impacts of fire and invasive species on desert soil ecology. *Rangel. Ecol. Manag.* **2011**, *64*, 450–462. [CrossRef]

9. Brooks, M.L.; Pyke, D.A.; Galley, K.; Wilson, T. Invasive plants and fire in the deserts of North America. In Proceedings of the Invasive Species Workshop: The Role of Fire in the Control and Spread of Invasive Species, Tall Timbers Research Station, Tallahassee, FL, USA, 19–22 June 2001.
10. Balloffet, N.; Deal, R.; Hines, S.; Larry, B.; Smith, N. 2012, Ecosystem Services and Climate Change, U.S. Department of Agriculture, Forest Service, Climate Change Resource Center. Available online: https://www.fs.usda.gov/ccrc/topics/ecosystem-services (accessed on 23 June 2023).
11. Canadian Wildland Fire Information System, Natural Resources Canada. Available online: https://cwfis.cfs.nrcan.gc.ca/report (accessed on 23 June 2023).
12. Russia Deforestation Rates & Statistics: GFW. Available online: https://www.globalforestwatch.org/dashboards/country/RUS (accessed on 26 June 2023).
13. Houghton, J. Global warming. *Rep. Prog. Phys.* **2005**, *68*, 1343–1403. [CrossRef]
14. Oceanic, N.; Administration, A. Available online: https://www.noaa.gov/noaa-wildfire/wildfire-climate-connection (accessed on 10 September 2023).
15. Brown, T.J.; Hall, B.L.; Westerling, A.L. The impact of twenty-first century climate change on Wildland fire danger in the Western United States: An applications perspective. *Clim. Chang.* **2004**, *62*, 365–388. [CrossRef]
16. Oh, K.S.; Jung, K. GPU implementation of neural networks. *Pattern Recognit.* **2004**, *37*, 1311–1314. [CrossRef]
17. Dawar, I.; Gupta, S.D.; Singh, R.; Kothari, Y.; Layek, S. Forest Fire Detection using Deep Learning Techniques. In Proceedings of the 2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN), Vellore, India, 5–6 May 2023; pp. 1–6.
18. Nallakaruppan, M.; Pillai, S.; Bharadwaj, G.; Balusamy, B. Early Detection of Forest Fire using Deep Image Neural Networks. In Proceedings of the 2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET), Warsaw, Poland, 19–21 May 2023; pp. 1–5.
19. Khan, A.; Hassan, B.; Khan, S.; Ahmed, R.; Abuassba, A. DeepFire: A novel dataset and deep transfer learning benchmark for forest fire detection. *Mob. Inf. Syst.* **2022**, *2022*, 1–14. [CrossRef]
20. Khan, S.; Khan, A. FFireNet: Deep Learning based forest fire classification and detection in smart cities. *Symmetry* **2022**, *14*, 2155. [CrossRef]
21. Supriya, Y.; Gadekallu, T.R. Particle swarm-based federated learning approach for early detection of forest fires. *Sustainability* **2023**, *15*, 964. [CrossRef]
22. Namburu, A.; Selvaraj, P.; Mohan, S.; Ragavanantham, S.; Eldin, E.T. Forest Fire Identification in UAV Imagery Using X-MobileNet. *Electronics* **2023**, *12*, 733. [CrossRef]
23. Idroes, G.M.; Maulana, A.; Suhendra, R.; Lala, A.; Karma, T.; Kusumo, F.; Hewindati, Y.T.; Noviandy, T.R. TeutongNet: A Fine-Tuned Deep Learning Model for Improved Forest Fire Detection. *Leuser J. Environ. Stud.* **2023**, *1*, 1–8. [CrossRef]
24. Alice, K.; Thillaivanan, A.; Koteswara Rao, G.R.; S, R.; Singh, K.; Rastogi, R. Automated Forest Fire Detection using Atom Search Optimizer with Deep Transfer Learning Model. In Proceedings of the 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), Salem, India, 4–6 May 2023; pp. 222–227.
25. Wu, S.; Zhang, L. Using popular object detection methods for real time forest fire detection. *arXiv* **2018**, *1*, 280–284.
26. Jin, S.; Lu, X. Vision-based forest fire detection using machine learning. In Proceedings of the 3rd International Conference on Computer Science and Application Engineering, Sanya, China, 22–24 October 2019; pp. 1–6.
27. Chen, G.; Cheng, R.; Lin, X.; Jiao, W.; Bai, D.; Lin, H. LMDFS: A Lightweight Model for Detecting Forest Fire Smoke in UAV Images Based on YOLOv7. *Remote Sens.* **2023**, *15*, 3790. [CrossRef]
28. Ghali, R.; Akhloufi, M.A. Deep Learning Approaches for Wildland Fires Remote Sensing: Classification, Detection, and Segmentation. *Remote Sens.* **2023**, *15*, 1821. [CrossRef]
29. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
30. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
31. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
32. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2017; pp. 1251–1258.
33. Mehta, S.; Rastegari, M. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv* **2021**, arXiv:2110.02178.
34. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
35. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
36. Mihalkova, L.; Mooney, R.J. Transfer Learning from Minimal Target Data by Mapping across Relational Domains. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-09), Pasadena, CA, USA, 11–17 July 2009; Volume 9, pp. 1163–1168.

37. O'Malley, T.; Bursztein, E.; Long, J.; Chollet, F.; Jin, H.; Invernizzi, L. KerasTuner. 2019. Available online: https://github.com/keras-team/keras-tuner (accessed on 19 October 2023).
38. Chollet, F. Keras. Available online: https://keras.io (accessed on 19 October 2023).
39. Mousavi, S.; Ilanloo, A. Nature inspired firefighter assistant by unmanned aerial vehicle (UAV) data. *J. Future Sustain.* **2023**, *3*, 143–166. [CrossRef]