Old Dominion University

ODU Digital Commons

Cybersecurity Undergraduate Research Showcase

2023 Fall Cybersecurity Undergraduate Research Projects

Integrating AI into UAVs

Huong Quach
Old Dominion University

Follow this and additional works at: https://digitalcommons.odu.edu/covacci-undergraduateresearch

Part of the Artificial Intelligence and Robotics Commons, Computer Engineering Commons, and the Theory and Algorithms Commons

Quach, Huong, "Integrating AI into UAVs" (2023). *Cybersecurity Undergraduate Research Showcase*. 16.
https://digitalcommons.odu.edu/covacci-undergraduateresearch/2023fall/projects/16

This Paper is brought to you for free and open access by the Undergraduate Student Events at ODU Digital Commons. It has been accepted for inclusion in Cybersecurity Undergraduate Research Showcase by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

Coastal Virginia Cyber Initiative

Department of Cybersecurity, Old Dominion University - Norfolk

Integrating Artificial Intelligence into Unmanned Aerial Vehicles

Huong Quach

Mentor: Peng Jiang

01 December 2023

Research Report Outline

Introduction

Artificial intelligence (AI), in the form of Machine Learning (ML), and Deep Learning(DL) are at the forefront of the technological revolution that is reshaping the way humans perceive and interact with the world around us. The field of artificial intelligence has become a driving force to many aspects of our daily lives, from devices that we use daily to industries that push the economy. This project applies DL techniques to develop a smoke detection algorithm for deploying on mobile platforms such as drones and other self-driving vehicles. The application is tailored for emergency response situations and aims to create and optimize AI algorithms that enhance the decision-making capabilities of these platforms.

The project was conducted in three phases: algorithm development, algorithm implementation, and testing and optimization. First a learning algorithm designed for mobile platforms with an emphasis on real-time decision making, obstacle detection, and offline computer vision navigation tailored for first responders in critical situations. The code was written in python in a Conda virtual environment using libraries including numpy, pytorch, torchvision, tensor. The algorithm was then integrated and implemented into the Jetson Nano platform, taking advantage of its cutting edge computing and AI capabilities, with the hope of providing more responsive support for first responders. Lastly the algorithm was tested and optimized on the Jetson Nano to ensure efficiency and effective performance.

Background

Drones and UAV Primer

Drones are unmanned aerial vehicles (UAVs) that can be piloted remotely or flown autonomously from predetermined missions, supplemented with real-time inputs from sensors, and deploying artificial intelligence algorithms to make decisions when contingencies occur. UAVs are so versatile that they are being deployed in many civil, commercial, and military applications. UAVs are used in such diverse applications as photography, cinematography, surveying, infrastructure inspections, search and rescue operations, perimeter patrol, and delivery services. And this is only the beginning of the potential drone applications still being discovered. No matter the purpose of the drones, they're commonly equipped with: sensors, cameras, and mission specific tools or payloads. They come in various shapes and sizes ranging from quadcopters to large fixed wing airframes.

Drone classification plays a pivotal role in understanding suitability for use in various missions and deployment of AI/DL algorithms on-board. Drones are classified based on size, flight endurance, and capabilities. These classifications include micro air vehicles (MAVs), nano air vehicles (NAVs), vertical take-off and landing vehicles (VTOL), low altitude short endurance), low altitude long endurance (LASE), medium altitude long endurance (MALE), and high altitude long endurance (HALE) [14]. There are also autonomy classifications for UAVs that range from no automation features to fully autonomous cooperative decision making. This translates to class I, II, III, IV, and V. Class I drones are controlled manually with no autonomous features. Class II and III drones are able to sense and alert. Class IV drones are capable of sense and avoidance, and class V drones also include fully autonomous capabilities [22]. An example of class II features are pre programmed flight paths and take-off and landing features. Similar to class II, class III drones also have pre-programmed flight features, but would also be able to detect obstacles and alert the remote pilot. Class IV drones are considered autonomous and don't need to be piloted, these drones are equipped with stacks of sensors and are capable of sensing its surroundings. Class IV drones have autonomy on board to manage contingencies while it completes its mission. Class V drones are completely autonomous and are able to fly under any conditions, however there are none currently in production.

The autonomy in class IV and V drones can be achieved to some extent using traditional programing methods. Traditional methods are stronger where deterministic output can be programmed based on a structured, and relatively small, input space - e.g. such as where the range of all inputs are known and the combinatorics of input values are known. When the input space is unstructured, the number of input signals is large, and their combination exponentially larger, deploying more complex detection and decision making capabilities on-board may require use of ML or DL algorithms. AI, specifically DL algorithms are particularly well suited for pattern matching of complex input space.

Trained DL models can be deployed in vision systems on board for detecting obstacles in ore near the flight path, people on the ground during landing, or used to derive localization information for validation with GPS localization information. AI/DL could also be deployed to detect cyber attacks in the form of spoofed navigation signals, communication links, and information coherence from off-board sources. DL can also be effectively deployed for decision making algorithms where the input space can not be totally described and the output space is described only in terms of an optimization objective. This third area has long been practiced in chess competitions [5] and more recently with the air force deploying AI/DL in wingman drones which help pilots in dog fights [11].

Artificial Intelligence Primer

Artificial intelligence is a broad term which encompasses machine learning, neural networks, and deep learning. Artificial intelligence is the development of computer systems that are capable of performing tasks that require human level intelligence; tasks that include learning, reasoning, problem solving, recognition, or translation. Artificial intelligence can be organized into two categories, weak and strong algorithms. Weak algorithms are designed to do specific tasks and are limited in scope, without learning capabilities. Strong algorithms are capable of human-like level cognitive abilities and can learn and apply knowledge [13].

Machine learning is a subset of AI which specializes in algorithms that learn and improve from continuous inputs and training. Essentially turning data into numbers and finding patterns in the data. Key components of machine learning are generic model creation, model training, testing, model evaluation, tuning, and learning. Machine learning can be achieved by several methods: supervised, unsupervised, and semi supervised or validation categories [9]. Supervised learning involves training algorithms to predict outputs based on labeled input samples. Unsupervised learning trains the algorithms based on unlabeled input samples, where the algorithm learns based on information absorbed from training data. Semi supervised learning is a mixture where some data is labeled and unlabeled.

A neural network is a subset of machine learning modeled after the human brain in the sense that the brain is composed of many billions of Neuron cells connected by Axon. Likewise, Neural Nets represent this form as a set of interconnected nodes and many layers. Neural network nodes are organized into layers which serve input, processing and output functions. The processing steps in a neural net algorithm are handled in the hidden layers which may consist of 150 or more processing steps before passing information to the output layer. The popular ResNet152,for example, implements 152 hidden layers performing image processing [15].

Each layer consists of many nodes which have weighted connections to nodes in the next layer. The input layer gathers and conditions the inputs before passing them to the nodes in the hidden layer(s). Each node in the hidden layers processes the inputs from the previous layer using its activation function. Activation functions are mathematical formulas which use the node's inputs to determine the output sent to the connected nodes in the next layer. The final hidden layer is connected to the output layer whose function is to translate results of the hidden layer to the original input context for the user to understand.

Neural networks (NN) are classified based on their architecture and data flow. There are numerous different types of neural networks used for different purposes and data types. Neural networks have been applied to image and speech recognition, forecasting, detection and diagnostic applications. The web site paperswithcode.com gives references to thousands of papers for computer vision and natural language procession which use some form of neural networks [18]. There is sometimes a distinction between NNs used for Machine Learning (ML) and Deep Learning. Table 1 and 2 below give notable ML algorithms in the two categories.

Table 1: Notable machine learning models.

AI Model Name	Brief Description	Additional Reference
Random Forest	Supervised learning algorithm used for classification and regression problems. Combines a number of decision trees.	What Is Random Forest? A Complete Guide | Built In
Gradient Boost	ML technique used typically in regression and classification tasks. Based on aggregating an ensemble of weak prediction models. Can be more accurate than Random Forest.	a) tremiller.dvi (su.domains) b) A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting (face-rec.org)
Naive Bayes	Group of linear probabilistic classifiers based on Bayes Theorem. Usually applied to large language models for language understanding	How Naive Bayes Algorithm Works? (with example and full code) | ML+ (machinelearningplus.com)
Nearest Neighbor	ML models are often included in classification or regression models. Takes advantage of the concept of nearest neighbor. Generates model output based on closest neighbors in the training data set.	K-Nearest Neighbor(KNN) Algorithm - GeeksforGeeks
Support Vector Machines	Supervised learning algorithm used for text and image classification. Also used in face and anomaly detection	Introduction to Support Vector Machines (SVM) - GeeksforGeeks

Table 2: Notable Deep Learning Models

AI Model Name	Brief Description	Additional Reference
Fully Connected NN	Type of NN where each node is connected to every node in the next layer.	a) O'reilly Publishers. TensorFlow for Deep Learning. Chapter 4. Fully Connected Deep Networks - TensorFlow

		for Deep Learning [Book] (oreilly.com) b) Fully Connected Layer vs Convolutional Layer: Explained | Built In
Convolutional NN	Typically used for object classification by vision. Typically good for processing spatial data. Tends to be size agnostic in classification results but has trouble with orientation.	a) What Is a Convolutional Neural Network? A Beginner's Tutorial for Machine Learning and Deep Learning (freecodecamp.org) b) How To Build And Train A Convolutional Neural Network | Nick McCullum
Transformer NN	Deep learning algorithms used for natural language processing.	a) The Ultimate Guide to Transformer Deep Learning (turing.com) b) Illustrated Guide to Transformers Neural Nets Bing Videos
Recurrent NN	Type of artificial NN (ANN) which has memory characteristics. Output of nodes are a function of current and previous inputs. Used for speech recognition applications	Recurrent Neural Network Tutorial (RNN) | DataCamp

Convolutional Neural Nets

This internship focuses on testing and tuning a vision processing algorithm for recognizing fire and smoke for first responder applications. Toward this end a Convolutional neural networks (CNNs) is used to build the DL algorithm. Wu gives a detailed introduction to the mathematics behind the CNN [32]. CNNs specialize in tasks involving visual data such as images and videos. A key feature of CNNs is scanning for patterns, which makes it very effective in image recognition applications. Key advantages of using CNNs are versatility, effectiveness for large data sets, adaptability with image sizes, and spatial hierarchies. Disadvantages of using CNNs are computational intensity and interpretability.

Convolutional neural networks consist of three main layers that are labeled: convolutional layer, pooling layer, and fully-connected layer. Each layer carries out spatial operations. The convolutional layer executes the majority of the computations. These operations identify visual elements in the input. This layer consists of input data, filters, and a feature map. Feature detectors are two dimensional arrays that represent sections of an image. Filters are generally three by three matrices but can vary in size, and determine the size of the receptive field. Filters are applied to sections of the image and the dot product is calculated between the input and filter values, and fed into an output array. This process repeats until the whole image is scanned. The final output array is called the feature map, or convolved feature [28].

Pooling layers usually come after the convolutional layer. The main purpose of these operations is to reduce computational cost and control overfitting [12]. Pooling layers downsample the dimensions of the data by reducing the parameters in the input and condensing the resolution. Essentially discarding irrelevant data. Popular pooling methods include average pooling, max pooling, stochastic pooling, spatial pooling, and region of interest pooling. Although much information is lost during the pooling process; complexity is reduced, efficiency is improved, and overfitting risk is reduced. The fully connected layer is the final layer of the convolutional neural network. It's fully connected in the sense that each node in the output layer is connected directly to a node in the preceding layer. Classification is dictated by this layer based on features filtered by the preceding layers.

Atienza gives a nice tutorial explaining the convolution, ReLu, Max Pooling functions in pictorial form. [7] And gives some treatment to the Receptive Field, impact of filter size on processing load, and use zero padding in output features.

Weights are modifiable numeric values that are triggered by inputs. Once an output is generated by the final layer the loss function is calculated and the back propagation process is performed; the weights are then adjusted accordingly to minimize loss [28].

Development Environment

The code development and testing for this internship was done in the PyCharm Integrated Development Environment (IDE). The code was executed in a Python virtual environment managed using the Conda package manager. The program uses libraries and modules including: python imaging library, pytorch and torchvision, numpy, matplotlib, torch autograd. These libraries and modules are essential for loading data sets, defining network architecture, setting up data transforms, training modules, and visualizing results.

The Pytorch library contains NN models which may be directly used to speed up the work of NN development. It also includes tools for processing, transforming image data, applying filters, adding text or shapes to images, and compressing or decompressing image data [21]. Pytorch and torchvision are the framework of the program that houses the neural network layers, computation, utils, data sets and transformers, optimizers, and pre-trained models [30]. Numpy is a mathematical library that specializes in numerical operations, mathematical functions, number generation, linear algebraic generation, and array multiplication [33]. Matplotlib is a plotting library that's great for data visualization, 2D/3D plotting, subplots, and in general high quality graphing [21]. The torch autograd provides functions that automatically differentiate, track gradients, and classify variables [3]. Most importantly, Pytorch supports NN execution in both the traditional CPU and parallel processing environment such as Nvidia's general purpose graphics processing units (GPGPU) [10].

Pythorch was originally developed at Facebook but later became public open source. Pythorch is currently used to build AI models at notable companies such as Tesla, OpenAI, Microsoft, and Meta. There are other ML/DL libraries suitable for this work – such as Keras, Matlab, Colab, TensorFlow. But Pytorch is, by far, the most used as indicated by published work in AI/ML/DL topics which contains code. Recent trends, published in the website "Papers With Code" [34], show that as much as 58% of technical work in this area is done with Pytorch. The pytorch documentation [3] is the best information source for developers.

Applications Objective

Fire and Smoke Model Used

The object of this application is to detect from a visual image if there is fire and smoke. This algorithm is tailored for first responders to deploy in critical situations such as aerial fire and smoke detection monitoring. The approach is to train a resnet50, a residual network of fifty layers that classify images into three categories: smoke, fire, and neutral. The algorithm could be deployed with the requisite hardware on an autonomous mobile platform for early detection.

Data Set Review

The data set used for this work consists of sixty-thousand images pulled from the world wide web and labeled into three categories: fire, smoke, and neutral. The neutral category includes images that are neither fire or smoke. The purpose of these categories is to teach the algorithm to recognize smoke, fire, and images that are neutral. Images are unstructured data, thus the program calls input conditioning functions to transform each input image to fit into specific size parameters.

Figure 1-3 gives examples of training images for the 3 classes. There is a variety of lighting contrast and distribution of fire and smoke as a percentage of the image size. The input for the Other class has similar contrast variations and also has similar colors to the fire and smoke training images.

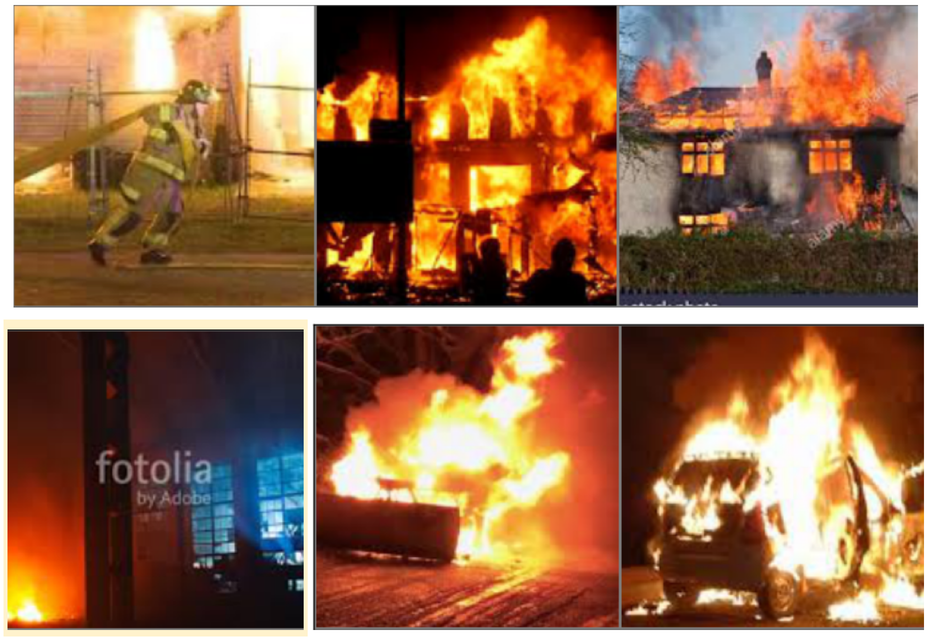


Figure 1: Sample images with fire for training the CNN

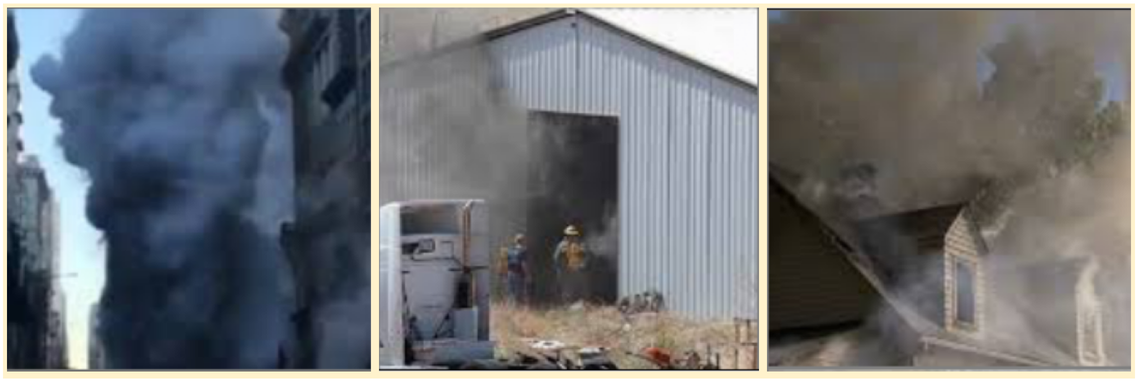


Figure 2: Sample of training images with only smoke.

Figure 3: Example of unrelated images for training to recognize smoke and fire.

Grouping inputs for Training

The input images are randomly shuffled so that each train epic uses a mixture of fire, smoke and neutral images. Each training epoch uses a batch of 64 images. The CNN model was trained over a specified number of epochs. Figure 4 gives an example set of images used in a training epoch.

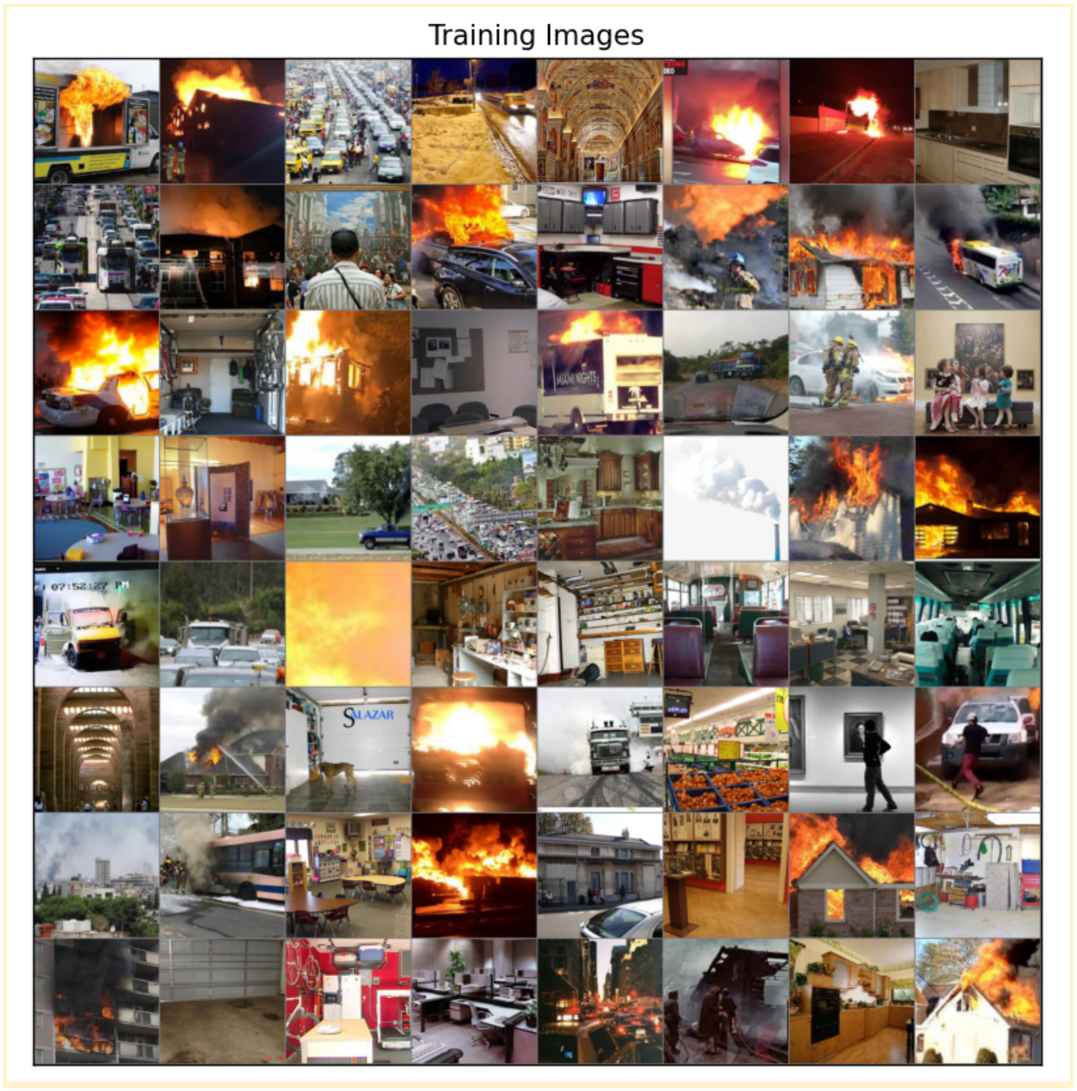
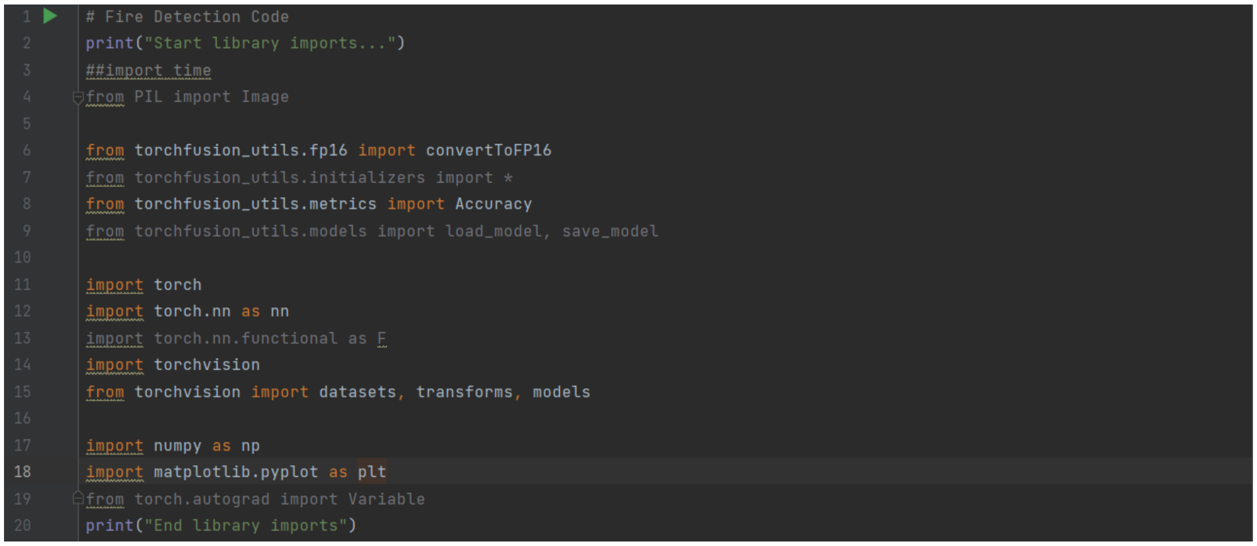


Figure 4: Sample set of 64 images grouped in one training epoch.

The Neural Net Code Walk-Thru

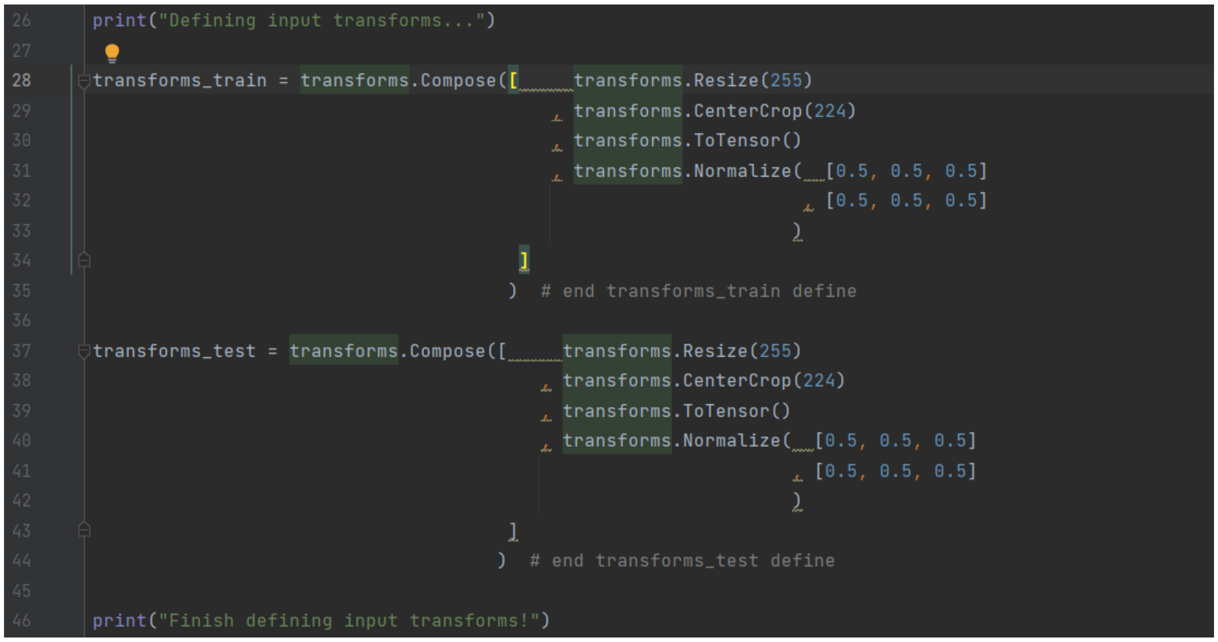
The neural network in this program is based on ResNet50, a residual network of fifty layers. ResNet50 is capable of classifying images into one thousand categories [25]. The program configures the ResNet to classify the input into 3 classes. These being Smoke, Fire, and Neutral. The reference code was obtained from the mentor. Key components of the network are: model architecture, device placement, loss function and optimizer, forward pass, learning rate scheduler, validation, and visualization. The pytorch workflow consists of data initialization, loading data into tensors, building a model to train, fitting model to data, evaluating model, improving through experimentation, then saving and reloading the trained model [31]. The general flow of the code is as follows below.

1. Import libraries

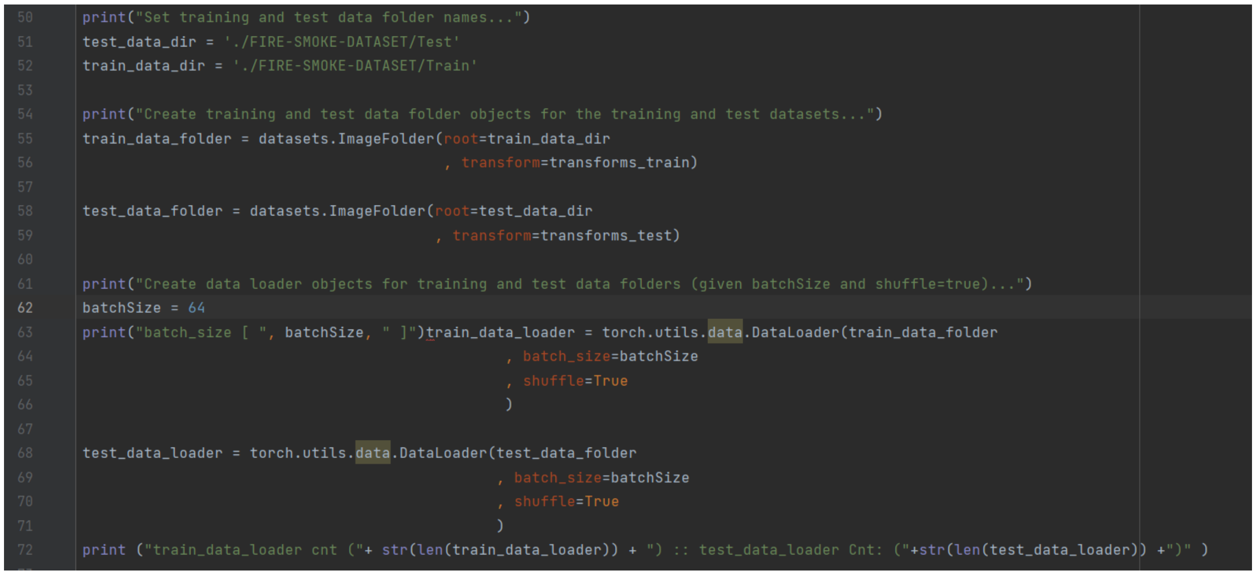


```
# Fire Detection Code
print("Start library imports...")
##import time
from PIL import Image

from torchfusion_utils.fp16 import convertToFP16
from torchfusion_utils.initializers import *
from torchfusion_utils.metrics import Accuracy
from torchfusion_utils.models import load_model, save_model

import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
from torchvision import datasets, transforms, models

import numpy as np
import matplotlib.pyplot as plt
from torch.autograd import Variable
print("End library imports")
```

2. data preparation begins: data directories, folders for testing and training data, and data loaders are implemented.

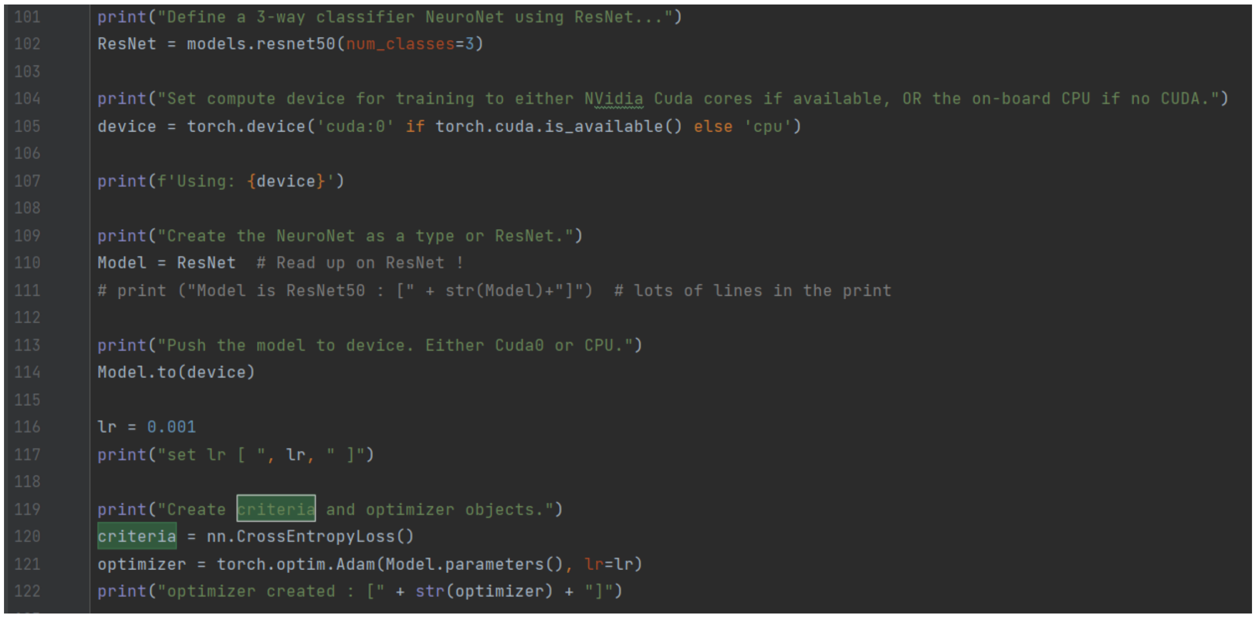


```
print("Defining input transforms...")

transforms_train = transforms.Compose([   transforms.Resize(255)
                                        , transforms.CenterCrop(224)
                                        , transforms.ToTensor()
                                        , transforms.Normalize(  [0.5, 0.5, 0.5]
                                                               , [0.5, 0.5, 0.5]
                                                               )
                                        ]
                                      )  # end transforms_train define

transforms_test = transforms.Compose([   transforms.Resize(255)
                                       , transforms.CenterCrop(224)
                                       , transforms.ToTensor()
                                       , transforms.Normalize(  [0.5, 0.5, 0.5]
                                                              , [0.5, 0.5, 0.5]
                                                              )
                                       ]
                                     )  # end transforms_test define

print("Finish defining input transforms!")
```



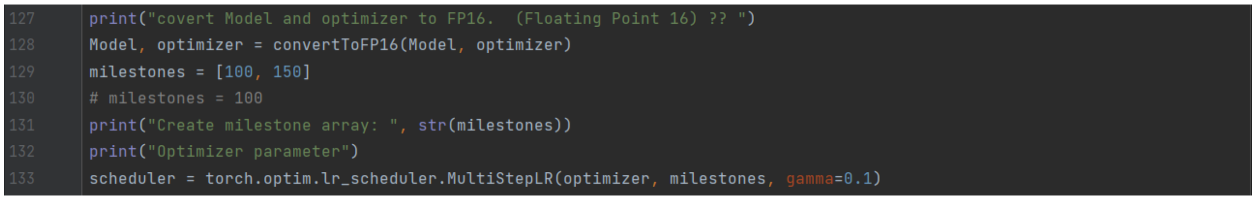
```
print("Set training and test data folder names...")
test_data_dir = './FIRE-SMOKE-DATASET/Test'
train_data_dir = './FIRE-SMOKE-DATASET/Train'

print("Create training and test data folder objects for the training and test datasets...")
train_data_folder = datasets.ImageFolder(root=train_data_dir
                                         , transform=transforms_train)

test_data_folder = datasets.ImageFolder(root=test_data_dir
                                        , transform=transforms_test)

print("Create data loader objects for training and test data folders (given batchSize and shuffle=true)...")
batchSize = 64
print("batch_size [ ", batchSize, " ]")train_data_loader = torch.utils.data.DataLoader(train_data_folder
                                                , batch_size=batchSize
                                                , shuffle=True
                                                )

test_data_loader = torch.utils.data.DataLoader(test_data_folder
                                               , batch_size=batchSize
                                               , shuffle=True
                                               )
print ("train_data_loader cnt ("+ str(len(train_data_loader)) + ") :: test_data_loader Cnt: ("+str(len(test_data_loader)) +")" )
```

3. The neural network model is then defined, pytorch has various neural networks. This algorithm defines a three way classification network categorized as smoke, fire, or neutral using the resnet50 model.



```
print("Define a 3-way classifier NeuroNet using ResNet...")
ResNet = models.resnet50(num_classes=3)

print("Set compute device for training to either NVidia Cuda cores if available, OR the on-board CPU if no CUDA.")
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

print(f'Using: {device}')

print("Create the NeuroNet as a type or ResNet.")
Model = ResNet  # Read up on ResNet !
# print ("Model is ResNet50 : [" + str(Model)+"]")  # lots of lines in the print

print("Push the model to device. Either Cuda0 or CPU.")
Model.to(device)

lr = 0.001
print("set lr [ ", lr, " ]")

print("Create criteria and optimizer objects.")
criteria = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(Model.parameters(), lr=lr)
print("optimizer created : [" + str(optimizer) + "]")
```

4. After that, device configuration is set. In this case, the virtual environment was configured using cuda interpreter.

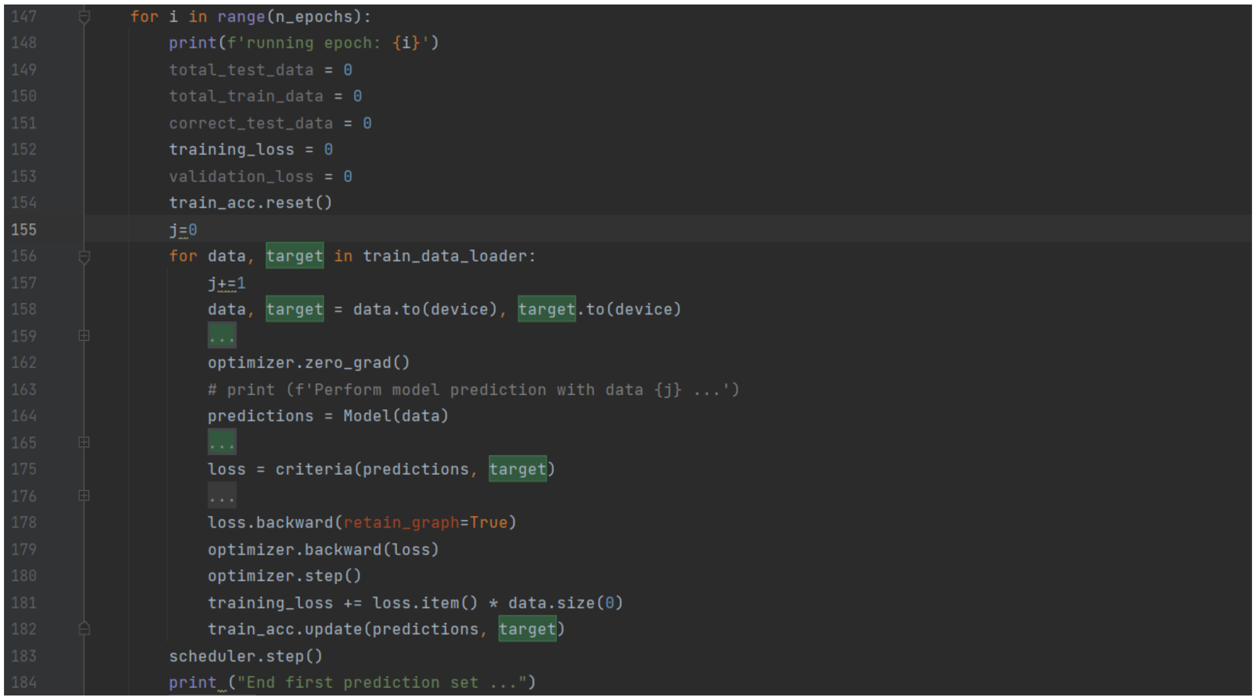


```
print("covert Model and optimizer to FP16.  (Floating Point 16) ?? ")
Model, optimizer = convertToFP16(Model, optimizer)
milestones = [100, 150]
# milestones = 100
print("Create milestone array: ", str(milestones))
print("Optimizer parameter")
scheduler = torch.optim.lr_scheduler.MultiStepLR(optimizer, milestones, gamma=0.1)
```

5. Model training and validation was then implemented and parameters such as loss function, optimizer, and learning rate were set up. The training and validation loops performed model training, and computed training and validation losses.

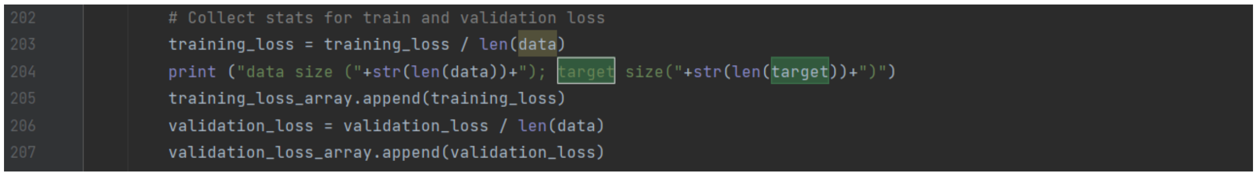
6. In the program the training optimizer used was Adam, cross enthalpy loss was used, with the learning rate set to 0.001.

7. The training loop performed forward pass, backward pass, calculated loss, optimization, and prints accuracy and loss for each batch.



```
for i in range(n_epochs):
    print(f'running epoch: {i}')
    total_test_data = 0
    total_train_data = 0
    correct_test_data = 0
    training_loss = 0
    validation_loss = 0
    train_acc.reset()
    j=0
    for data, target in train_data_loader:
        j+=1
        data, target = data.to(device), target.to(device)
        ...
        optimizer.zero_grad()
        # print (f'Perform model prediction with data {j} ...')
        predictions = Model(data)
        ...
        loss = criteria(predictions, target)
        ...
        loss.backward(retain_graph=True)
        optimizer.backward(loss)
        optimizer.step()
        training_loss += loss.item() * data.size(0)
        train_acc.update(predictions, target)
    scheduler.step()
    print ("End first prediction set ...")
```

8. The validation loop runs through batches of validation data and calculates loss and accuracy.



```
    # Collect stats for train and validation loss
    training_loss = training_loss / len(data)
    print ("data size ("+str(len(data))+"); target size("+str(len(target))+")")
    training_loss_array.append(training_loss)
    validation_loss = validation_loss / len(data)
    validation_loss_array.append(validation_loss)
```

Initial training results were plotted and given in Figure 5. After 30 training epics, the model had an accuracy of about 98% with a training loss about 2.5. (Training loss: 2.4895458984375, Tran_Accuracy: 0.978723406791687) The training loss plotted in Figure 5 shows a large drop within the first 5 epics. And after that, the incremental improvements between epoch was much smaller compared to the first 5 epochs.

Training Initial Results

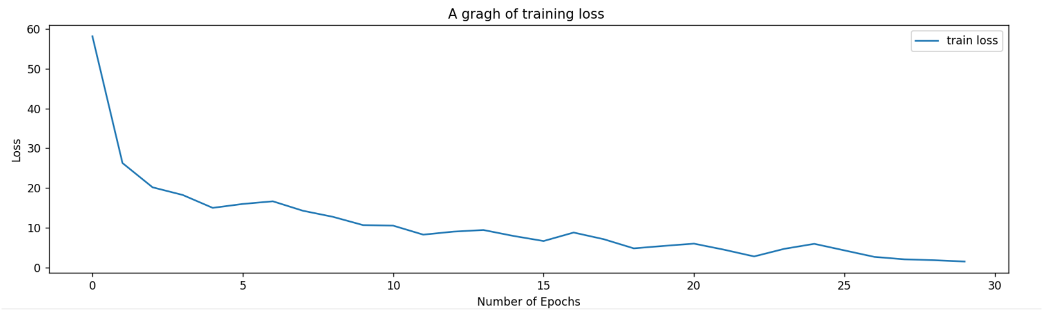
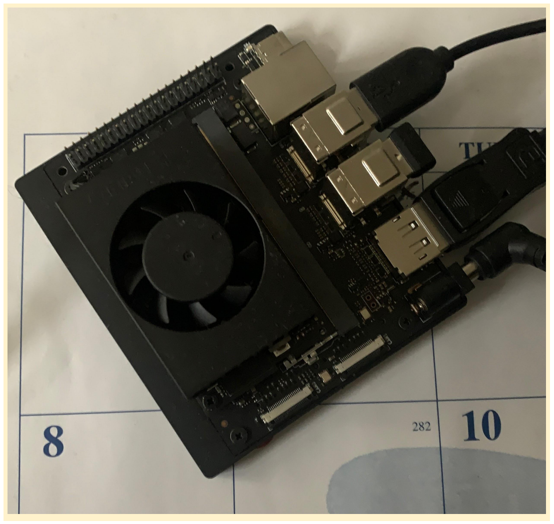


Figure 5: Results from training the ResNet50 model using 1950 image training set.

Jetson Nano

The Jetson Nano Is a small single board computer (SBC) designed by NVIDIA. It's designed for embedded systems and tasks tailored for artificial intelligence. The Nano is used in a number of autonomous vehicle designs. The SBC (Figure xxx) is equipped with: a 1024 core GPU with 32 tensor cores, 64-bit CPU, microSD card slot, 8 gigabytes of LPDDR5. Along with: a display port, four USB type A ports, and a USB type C port (*NVIDIA Jetson Nano*, n.d.).

The Nano (Figure 7) used on this project was brand new and required loading an operating system (OS) image. The procedure for creating a bootable image was followed. This included the following steps: unzipping the OS image downloaded, downloading an SD writer, burning a copy of bootable OS onto the microSD, and inserting the microSD into the nano. After a series of bios option selections, the nano booted flawlessly. A set of software tools were downloaded from the internet including: python and pycharm. The pytorch libraries were then installed.


.
Figure 7: Jetson Nano used for this internship

The bulk of the CNN exploration was performed on the NVidia laptop provided by the mento. The code was also tested on the Jetson Nano. Figure 8 shows the screens of the Fire-Smoke classifier running. There is still some remaining configuration work needed to run the code in the Nano's Cuda cores.

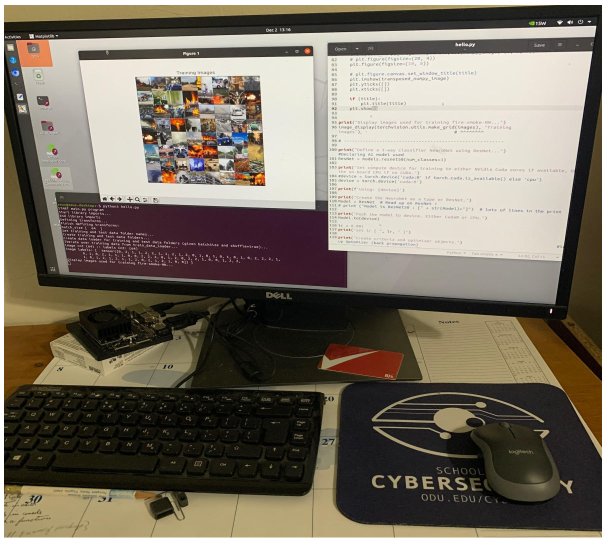


Figure 8: Jetson Nano configured and running the Fire-Smoke Classifier CNN.

Acknowledgements

The completion of this paper has been made possible through the Coastal Virginia Cyber Initiative. I’m grateful for the opportunity to learn more about artificial intelligence and gain research experience. I would like to thank Dr. Peng for mentoring me throughout the semester, providing access to equipment and material to complete this project, and setting time aside for meetings.

References

[1] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)* (pp. 1-6). Ieee. https://ieeexplore.ieee.org/abstract/document/8308186

[2] Alelyani, S. (2021). Detection and Evaluation of Machine Learning Bias. Applied Sciences, 11(14), 6271. https://doi.org/10.3390/app11146271

[3] Automatic differentiation package - torch.autograd — PyTorch 1.10.0 documentation. (n.d.). Pytorch.org. https://pytorch.org/docs/stable/autograd.html

[4] Basha, S. H. S., Dubey, S. R., Pulabaigari, V., & Mukherjee, S. (2019). Impact of fully connected layers on performance of convolutional neural networks for image classification. Neurocomputing. https://www.sciencedirect.com/science/article/abs/pii/S0925231219313803

[5] Bloomfield, B. P., & Vurdubakis, T. (2008). IBM's Chess Players: On AI and Its Supplements. The Information Society, 24(2), 69–82. https://doi.org/10.1080/01972240701883922

[6] B. M. Wilamowski, "Neural network architectures and learning algorithms," in IEEE Industrial Electronics Magazine, vol. 3, no. 4, pp. 56-63, Dec. 2009, doi: 10.1109/MIE.2009.934790.

[7] CNN Models: Concepts, Building and Training using PyTorch. (n.d.). Www.youtube.com. Retrieved December 1, 2023, from CNN Models: Concepts, Building and Training using PyTorch

[8] Confusion Matrix - an overview | ScienceDirect Topics. (n.d.). Www.sciencedirect.com. https://www.sciencedirect.com/topics/engineering/confusion-matrix#:~:text=A

[9] El Naqa, I., & Murphy, M. J. (2015). What Is Machine Learning? Machine Learning in Radiation Oncology, 3–11

https://doi.org/10.1007/978-3-319-18305-3_1

[10] FreeCodeCamp.org. (2022). PyTorch for Deep Learning & Machine Learning – Full Course. In YouTube. https://www.youtube.com/watch?v=V_xro1bcAuA

[11] Fish, T. (2022, November 1). Uncrewed ambitions of the Loyal Wingman. Airforce Technology. https://www.airforce-technology.com/features/uncrewed-ambitions-of-the-loyal-w

[12] Gholamalinezhad, H., & Khosravi, H. (2020, September 16). Pooling Methods in Deep Neural Networks, a Review. ArXiv.org. https://doi.org/10.48550/arXiv.2009.07485

[13] Glover, E. (2022, September 29). Strong AI vs Weak AI: What's the Difference | Built In. Builtin.com. https://builtin.com/artificial-intelligence/strong-ai-weak-ai

[14] Hassanalian, M., & Abdelkefi, A. (2017). Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, *91*, 99–131. https://doi.org/10.1016/j.paerosci.12017.04.003

[15] Hayou, S., Clerico, E., He, B., Deligiannidis, G., Doucet, A., & Rousseau, J. (n.d.). Stable ResNet. Retrieved November 26, 2023, from https://proceedings.mlr.press/v130/hayou21a/hayou21a.pdf

[16] IBM. (2021, March 4). The Neural Networks Model. Www.ibm.com. https://www.ibm.com/docs/en/spss-modeler/18.0.0?topic=networks-neural-model

[17] IBM. (2023). What are Convolutional Neural Networks? | IBM. Www.ibm.com. https://www.ibm.com/topics/convolutional-neural-networks

[18] Krabbenhöft, H., & Barth, E. (n.d.). TEVR: IMPROVING SPEECH RECOGNITION BY TOKEN ENTROPY VARIANCE REDUCTION. Retrieved December 2, 2023, from https://arxiv.org/pdf/2206.12693v1.pdf

[19] Krogh, A. (2008). What are artificial neural networks? Nature Biotechnology, 26(2), 195–197. https://doi.org/10.1038/nbt1386

[20] Marr, B. (n.d.). The Problem With Biased AIs (and How To Make AI Better). Forbes. Retrieved November 30, 2023, from https://www.forbes.com/sites/bernardmarr/2022/09/30/the-problem-with-biased-

[21] Matplotlib Tutorial. (n.d.). Www.w3schools.com. https://www.w3schools.com/python/matplotlib_intro.asp

[22] McNabb, M. (2019, March 11). DRONEII: Tech Talk – Unraveling 5 Levels of Drone Autonomy. DRONELIFE. https://dronelife.com/2019/03/11/droneii-tech-talk-unraveling-5-levels-of-drone-autonomy/

[23] NVIDIA Jetson Nano. (n.d.). NVIDIA. Retrieved December 2, 2023, from https://www.nvidia.com/en-in/autonomous-machines/embedded-systems/jetson-n

[24] Oken, A. (2017). An Introduction To and Applications of Neural Networks. https://www.whitman.edu/Documents/Academics/Mathematics/2017/Oken.pdf

[25] ResNet-50 convolutional neural network - MATLAB resnet50. (n.d.). Www.mathworks.com. https://www.mathworks.com/help/deeplearning/ref/resnet50.html

[26] Shafique, A., Mehmood, A., & Elhadef, M. (2021). Survey of Security Protocols and Vulnerabilities in Unmanned Aerial Vehicles. *IEEE Access*, *9*, 46927–46948. https://doi.org/10.1109/access.2021.3066778

[27] Structured vs Unstructured Data: What's the Difference? (2020, August 26). MonkeyLearn Blog. https://monkeylearn.com/blog/structured-data-vs-unstructured-data/#:~:text

[28] Team, G. L. (2020, April 29). Types of Neural Networks and Definition of Neural Network. GreatLearning. https://www.mygreatlearning.com/blog/types-of-neural-networks/

[29] The back propagation method for CNN | IEEE Conference Publication | IEEE Xplore. (n.d.). Ieeexplore.ieee.org. Retrieved November 25, 2023, from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=409626

[30] Torchvision — Torchvision master documentation. (n.d.). Pytorch.org. https://pytorch.org/vision/stable/index.html

[31] Tutorial. (n.d.). Pillow.readthedocs.io. https://pillow.readthedocs.io/en/stable/handbook/tutorial.html

[32] Wu, J. (2017). Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, *5*(23), 495. https://cs.nju.edu.cn/wujx/paper/CNN.pdf

[33] W3Schools. (2022). Introduction to NumPy. Www.w3schools.com. https://www.w3schools.com/python/numpy/numpy_intro.asp

[34] *Zapers with Code - Papers With Code : Trends*. (n.d.). Paperswithcode.com. Retrieved December 2, 2023, from https://zaperswithcode.com/trends