https://doi.org/ 10.33472/AFJBS.6.10.2024.4406-4415

African Journal of Biological Sciences

Journal homepage: http://www.afjbs.com

Research Paper Open Access

Using Dilated CNN and MLOPS for PII Detection in Pharmaceutical Reporting

Raj Kumar Keshri[1] Amit Chakraborty[2] Saptarshi Das[3] Chirantana Mallick[4]

1,2,3,4 JIS Institute of Advanced Studies and Research

Article History
Volume 6,Issue 10, Feb 2024
Received:28 Apr 2024
Accepted : 25 May 2024

doi: 10.33472/AFJBS.6.10.2024.4406-4415

Abstract

Personally Identifiable Information (PII) identification, redaction, and de-classifying if necessary is of prime importance not just to meet several compliance standards for healthcare, banking, insurance, and other business lines but also to safeguard and protect consumer interest. There exist methods to extract PII from textual corpus but for images there is scant of easily trainable and deployable models. In this paper we have explored the use of transfer learning combined with dilated convolution in a Convolution Neural Network (CNN) to find the segments of an image which corresponds to a PII looking field and demarcate it. The segment then identified is classified in a different color apart from a body color map that is pre-configured for our images. The training set consists of original images and masked segments that represent PII. The test set consists of newly obtained and invoice images that contains PII information in same spatial coordinates. The network built also contains up sampling to modify the convolution lenses that are used to visualize the field of interest of the image i.e., the segments so that the field of vision is broadened. Serverless Architecture is an emerging trend in modern cloud computing that is becoming increasingly popular due to its flexibility, scalability, cost efficiency, and agility. It allows organizations to focus on their core business objectives and development rather than managing their own servers. This paper focuses on the use of Microsoft Azure Functions [10] and Continuous Integration and Continuous Deployment (CICD) Pipelines for deploying serverless architectures. It discusses the benefits of this approach, including cost savings, scalability, and agility. The paper also examines the challenges associated with this approach, such as security and cost, as well as how to overcome these challenges

Introduction

Personally Identifiable Information (PII) redaction is mandatory for complying with various compliance standards and privacy/security laws – nearly all business domains and verticals must cater to several compliance standards that deals with ways to handle PII data. An example is HIPAA compliance which catalogs methods to deal with PII specially with all those data that can lead to a particular individual from any textual or image related data. [11]

PII redaction comes as the first step in anonymizing data and then mask it [11] – To redact the data there is a challenge to first identify the same and then have a bidirectional way to redact the same.

To address the above with the existing informational systems following are some of the challenges:

- While solutions to redact PII exists, the most popular of them uses regular expressions and hence are not suitable to redact from PII from texts containing variable formats
- Existing solutions also doesn't encompass custom PII information that is domain support for PII redaction, for e.g., an information can be a PII for healthcare projects and not a PII for financial domain
- Readily integrating a PII redaction solution in an existing product, developers should be able to train the PII redaction solution conveniently specific to their domain
- No standard way to measure the effectiveness of the solution

In this paper we have used transfer learning in a standard CNN network called Deep Lab and added some additional dense layers consisting of dilated convolutions on top of that to try segment-colored images. The images that we have used for segmentation are mostly public domain available invoice images that consists of sample personal information. We have first segmented the training images using regional segmentation so that the PII fields within the training images form a segment that can be readily used as a training data set. Hence the training data set consists of two folders: the non-segmented images of the invoices and the segmented ones that contains the segmentations across the PII. The CNN with the dilated layer thus constructed is then trained with the images. The CNN that is constructed consists of a dilation layer in its dense layer structure which enhances its field of view and increases the pixel size of field. The neural network top layer is up sampled to concentrate more on the regions where the pre-determined regions of segmentation has been identified. In this step the spatial dimensions of the PII regions are transformed as equal to the spatial dimensions of the input image. It is an operation that can be thought of a reverse of the pooling layer. It is of the type of a backwards convolution so that the areas of segments are specifically focused and learnt. This can also be used as a transposed CNN layer. We have provided the architecture of the CNN that we have used below. The full layers of the CNN are skipped and only the upper layer of the architecture is focused because of the uniqueness of the same as it is used to learn the segments of interest from the training images. Doubling the convolution matrix doubles the size of the lens that we use to view the segmented portions of the images. Once the CNN is trained, we have tested it with new invoice images to point out or segment the areas of interest that is the areas where there is PII, and we have found the network to be almost 79% accurate. While this is not a good accuracy the network thus constructed can be used to segment images which has multiple color regions and need not have the need to segment on basis of fixed length and stratified image patterns. A future work has also been identified in the form of using Recurrent neural networks with Convolution Neural Networks as an alternative for segmentation of images to identify Personally Identifiable Information (PII).

Serverless Architecture

Serverless architectures have become increasingly popular in the software industry due to their ability to reduce the cost of development and maintenance, while also increasing the scalability and reliability of cloud-hosted applications. The idea behind serverless architectures is that the underlying infrastructure is abstracted away from the developer, so that the developer can focus on the application logic instead of worrying about the infrastructure. Azure Functions is a serverless compute service that enables developers to run code without having to manage a server or virtual machine. It is a cloud-based platform that allows developers to easily create and deploy applications using a range of programming languages. Additionally, Azure Functions can be deployed using a Continuous Integration and Continuous Delivery (CI/CD) pipeline [3].

Personally Identifiable Information (PII) and compliance standards

From a business perspective also it is of utmost importance to detect, identify (read classify) and redact personally identifiable information. PII not only can be reasonably identified directly but indirectly also it can be associated uniquely with a person or a group of person to be identified individually. Cross referencing is also by all means a defined PII routing through which a person or a group of person can be identified uniquely. The identifiable information can further be classified as sensitive, confidential or high risk data. Multiple laws that deals with how a PII should be treated in a business system has been formulated which in turn has given rise to compliance standards that helps architects and other stakeholders of a business system to keep in track of how to use, redact and deal with PII in their respective business systems. Data storage, querying and deletion should follow these compliances

specifically. Here we have explored some popular compliance in context with the health care and banking and finance domain:

Health Insurance portability and Accountability Act (HIPAA) – It began in 1996 but is revised every year. It is used to protect personally identifiable information from a patient perspective that is not limited to the sensitive information like name, age, address etc. of the patient but also PII like the disease and treatment etc. HIPAA compliance is essential for any system that is dealing with health care in the USA but likewise system of compliance also exist in other parts of the world. The parties that are covered with such compliance requirements are health care providers, claim agents, clearinghouses and agents so that the claim processing, data analysis review and billing all fall under such compliance.

Sarbanes Oxley Compliance – Basic controls and Information technology controls falls under the SOX compliance but it maintaining the controls it is extremely important in all respect to take care of the Personally identifiable information. This means the audit trail and other necessities associated with managing the compliance has to support the tracking, identifying and redacting PII on a need to need basis. To safeguard data and to maintain the PII alongside there is a wide need to use this type of compliance with an automated process of tracking, redacting and then de classifying (on a need basis) the PII from every transaction that is recoded.

Apart from these two very important and impactful compliances of health care and financial domain there are many others in other domains in which it is fundamentally important to take care of PII. The importance of taking care of PII is immense and can be overwhelming even with manual teams supporting.

Image Segmentation

As we saw above classification is through which we can classify images in categories. Object detection helps us detect objects in an image and infers the coordinate of the object by using coordinate representation in a bounding box. Using these two techniques we derive image segmentation and semantic or instance segmentation for this business problem. It assigns a label to every pixel in an image so that the same labelled pixel in an image shares some specific characteristics [2]. It makes the image representation more meaningful and much easier to analyze. In semantic segmentation against an input image the trained neural network predicts the exact position of the object. The difference between object detection and image segmentation is that in the later its "pixel precise" [3], i.e., the pixels of the object of interest is precisely detected by the neural network. Image segmentation can be of two different types:

Semantic segmentation – In semantic segmentation the deep learning model tries to separate out the different pixels in the image according to the image representation of the same [4]. The segmentation is applied in the overall image to constitute the free classes of the image. Semantic segmentation does not distinguish (classify) between the objects in the image.

Instance segmentation – This is like semantic segmentation except that it will classify the pixel according to their class as well as marks the objects clearly if they all belong to the same class [5]. For e.g., if there are two people in the same image, this kind of segmentation will classify them as person 1 and person 2 along with marking their pixels.

Multiclass image segmentation with dilated CNN

Now let's focus on multi-class image segmentation and a pseudo code solution for the same. As seen in preceding section segmentation is one of the prime methods for object detection also. In this section we will focus on how to segment an image according to the presence of a fixed length string (such as an Aadhar number or a SSN number). We will also see how to apply the trained model to segment sections of a new image that contains a similarly format. The same novel method can apply to other precision image segmentation tasks. [6]

Image and its ground truth – Ground truths are also known as mask, which is a multiclass image derived from the parent image that separates the parent image into two or more regions. The different segments in an image thus derived usually can be mapped to separate classes in an image [7]. This is the reason why we call it as multi class segmented mask. Multiclass image segmentation process is divided into two steps: Training and Inference. The original images and corresponding ground truth are fed to a neural network (read RCNN) and trained to obtain a .h5

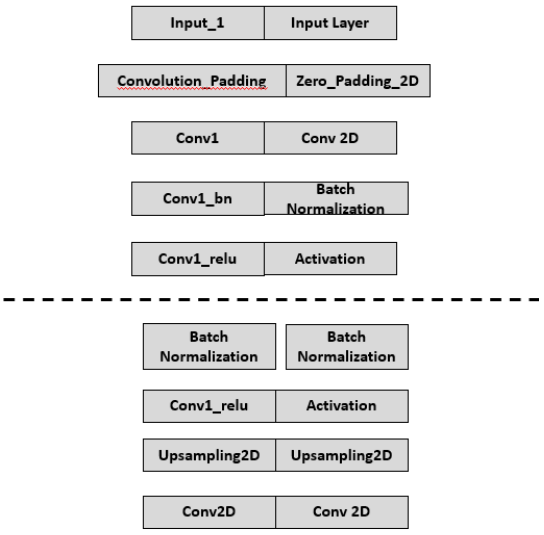
file as the trained model. The inference is obtained by feeding a new image into the trained model and getting the segmented masks and their classes.

Convolution Neural Network with Dilated layer with transfer learning

A dilated Convolution Neural Network is one in which the input kernels have are deliberately infused with missing elements to create dilation holes between the consecutive elements of the network. It involves pixel skipping so that a larger area of the image in which the convolution operations are being made can be covered [8]. The parameter of the network, d, is used to denote the dilation factor of the network. It helps to magnify the area of the input image and thus the segments within it in order to obtain information that can't be obtained with a normal pooling and convolution operation. We have used this along with transfer learning from Deep Lab that is a state of art semantic image segmentation model devised by Google. It is a deep learning model that uses encoder-decoder architecture to evaluate precise object boundaries and then recover the same through the convolve layers. Atrous Convolution layer is applied in the encoder. Atrous Convolution is a convolution architecture that consists of two parts: Depth wise convolution and point wise convolution [9]. Depth wise convolution layer applies convolution filters to each input channels while point wise convolution summations the output of depth wise convolution channels. The decoder architecture is primarily used for up sampling to get back the image boundaries from the encoded convolutional output [10]. A .h5 file contains the segment weights obtained after training the images and their corresponding ground truths. It supports complex heterogeneous data. A human color map file is also used to contain the color code for classifying the different segments thus obtained.

Model Architecture

Below is the model architecture thus prepared



The pseudocode for segmenting images using a dilated CNN is as follows:

*Resize image () --> module to resize image to 512*512 pixel:*

Image = Image.decode(image, channel =1) #normalize the image, i.e., every pixel is colored from -1 to +1 making every image uniformly distributed across the total loss

Image = Image.set_shape([None, None, 1])

*Image = Image.resize(512,512) #setting the image size uniformly to 512*512 pixels*

Batch data () --> module to divide the train and test data set to batches

DeepLab architecture specification

Number_of_filters, Kernel_size, Dilation_rate, paddings are set.

Initializer = HeNormal #For drawing samples from truncated normal distribution

Dilated Spatial Pyramid Pooling #encode multi-scale contextual information

Setup Average pooling
Setup Convolution Block
Setup the output layers for the encoding layers
DeepLabV3Plus #inputs images and output segmentation masks with specified number of classes
Setup RESNET with weights = "imagenet", include_top=false
Setup "top" layer
Upsample TOP layer with size = image_size/4 for both X and Y axis and interpolation = "bilenear"
NUM_CLASSES = 20
Model Compilation
Loss = Sparse Categorical Cross Entropy (Logistic Loss = True) #We used crossentropy loss between the training images and labels and masked image labels
Optimizer = "Adam"
Learning_Rate = 0.001
IF Early_Stopping == 1
Minimize (Loss) <= Loss_Threshold
Save Model Checkpoint (Checkpoint.h5)

Note:

- The tensor_slices method is used in the actual implementation to get the images from the original (now normalized) folder and mask folder as objects.
- Num_parallel_call method is used to specify how many parallel cores are to be used for the training and testing purpose

Training the model and Results

The model training result is as under (sample 10 epochs):

Epoch	Val_Loss	Val_Accuracy
1	3.5168	0.6535
2	2.3794	0.7034
3	1.0821	0.7687
4	0.9484	0.7197
5	0.8497	0.7468
6	0.7973	0.7452
7	0.9947	0.7703
8	0.9202	0.7836
9	1.1892	0.7834
10	1.0370	0.7687

- Most Optimized Val_Loss = 0.7973
- Maximum Val_Accuracy = 0.7836

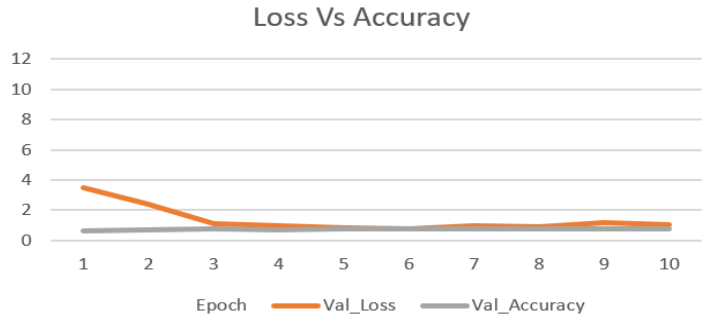


Fig 2: Loss Vs Accuracy (per epoch)

Maximum accuracy we could achieve for segmentation using dilated CNN is: 78.36 %

Serverless Functions with Automated Deployment via DevOps

The introduction of serverless functions and automated deployment via DevOps has made it much easier for developers to quickly develop, test and deploy web applications [6]. Serverless functions require a significant amount of configuration and are difficult to debug. Automated deployment via DevOps allows deploying applications and services to production environments using automated tools and processes [3]. It helps to ensure that the code is deployed quickly and reliably, and allows for faster development and deployment cycles. However, it also requires significant knowledge of the underlying infrastructure and can be difficult to manage. Recent research has explored solutions such as continuous integration/deployment (CI/CD) pipelines [8]. These solutions provide automated deployment, scalability and cost savings. Additionally, they allow for the integration of serverless functions with existing infrastructure and services. Overall, serverless functions and automated deployment via DevOps are powerful tools that can help developers deploy web applications quickly and reliably.

The proposed deployment method

Architectural patterns for public cloud, with special focus on Microsoft Azure, are becoming increasingly important as more businesses move their operations to the cloud. Microsoft Azure provides a range of cloud services, including compute, storage, database, and networking, to help businesses of all sizes build, deploy, and manage applications in the cloud. For organizations looking to take advantage of the cloud, Azure offers a comprehensive set of features, including the ability to easily scale resources up and down, high availability, cost-effectiveness, security, and more. Architectural patterns for public cloud, with special focus on Microsoft Azure, are designed to help organizations design and build cloud-based applications that meet their specific needs. In addition, architectural patterns for public cloud with special focus on Microsoft Azure can help organizations identify the best practices for deploying and managing their applications. A key component of any architectural pattern is the design of the application's architecture. This involves identifying the components that make up the application and how they interact with each other. For example, an application may use Azure services such as Azure Storage, Azure Service Bus, and Azure Web Apps. The next key component of any architectural pattern is the design of the application's infrastructure. This involves deciding which resources will be used to host the application, such as Azure Virtual Machines, Azure App Services, and Azure Databases [15]. It also involves selecting the appropriate services and features for the application, such as the authentication, authorization, and encryption. This is a critical step as it ensures that the application is secure, reliable, and scalable. The final component of an architectural pattern for public cloud with special focus on Microsoft Azure is the deployment process. This involves deploying the application to the cloud, as well as configuring it to work with the appropriate services and features. This involves setting up the necessary security settings, configuring the environment, and deploying the application to the cloud. The deployment process should be automated, as this ensures the application is up and running quickly. In summary, architectural patterns for public cloud with special focus on Microsoft Azure provide organizations with a way to design and build cloud-based applications that meet their specific needs [16]. They provide a way for organizations to define the components of their application and how they interact with each other, in order to ensure the application functions properly and provides the desired features and capabilities. Additionally, they provide organizations with a way to design and deploy their applications to the cloud and configure them to work with the appropriate services and features.

Methodology to Create and Deploy Azure Function

Start
Step 1: Create an Azure Function
Step 2: Configure the Trigger
Step 3: Configure the Function App Settings
Step 4: Create the Code for the Function

Step 5: Deploy Your Code
Step 6: Monitor Performance
Step 7: Troubleshoot, if Necessary.
End

Methodology to Create Azure DevOps CICD Pipeline

Start
Step 1: Create build and release pipelines
Step 2: Set up source control
Step 3: Configure build pipeline
Step 4: Configure release pipeline
Step 5: Configure environments
Step 6: Add deployment steps
Step 7: Configure triggers
Step 8: Run build and release pipelines
Step 9: Monitor and troubleshoot
End

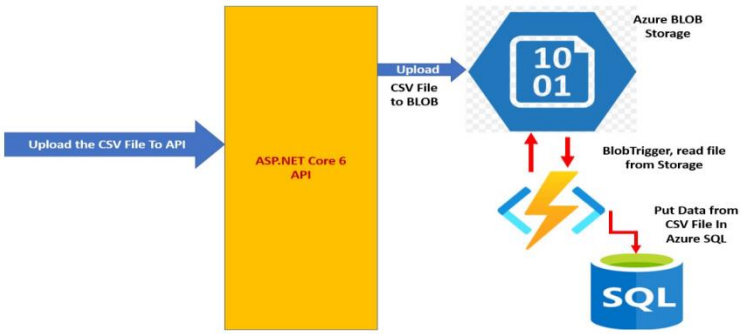


Figure 1: System Architecture Design

4.8 Sample CI pipeline. yml

azure-pipelines.yml

Contents History Compare Blame

```yaml
# Function app name
functionAppName: 'AzFnCDC'

# Agent VM image name
vmImageName: 'windows-2019'

# Working Directory
workingDirectory: '$(System.DefaultWorkingDirectory)/ServerlessArch'

stages:
- stage: Build
  displayName: Build stage

  jobs:
  - job: Build
    displayName: Build
    pool:
      vmImage: $(vmImageName)

    steps:
    - task: DotNetCoreCLI@2
      displayName: Build
      inputs:
        command: 'build'
        projects: |
          $(workingDirectory)/*.csproj
        arguments: --output $(System.DefaultWorkingDirectory)/publish_output --configuration Release

    - task: ArchiveFiles@2
      displayName: 'Archive files'
      inputs:
        rootFolderOrFile: '$(System.DefaultWorkingDirectory)/publish_output'
        includeRootFolder: false
        archiveType: zip
```

Figure 2: CI Pipeline Yaml file

Automated CI pipeline Triggered

As soon as any push is made on the Configured triggered Repo, the continuous integration i.e., CI pipeline gets automatically triggered and run the configured steps in an automated manner. Individual steps are executed in sequence to ensure the previous steps are successfully completed and produces the expected results.

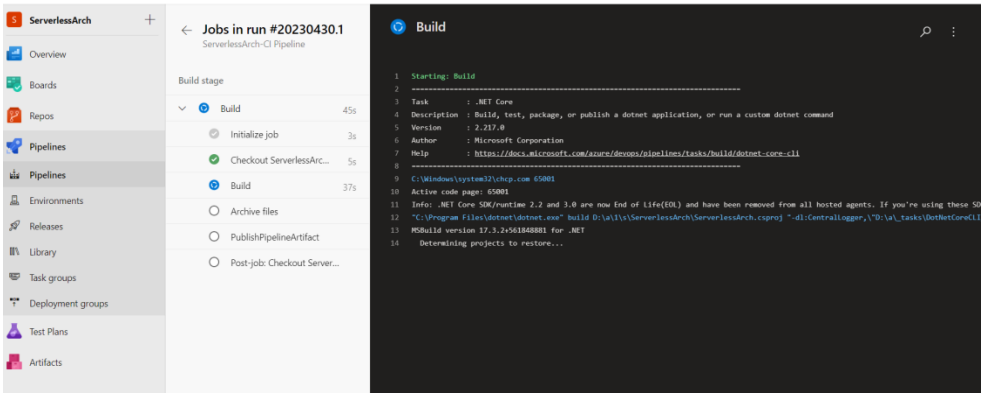


Figure 3:CI Build pipeline

CI pipeline completes and Artifacts Created

The automatically triggered continuous integration i.e., CI pipeline upon successfully completion creates an artifact, which holds all the required deployable file and the dlls, in a zip file. This Artifact ensures that the next steps that i.e., the continuous deployment (CD) pipeline has all the required files with the help of which it can deploy to the target resource.

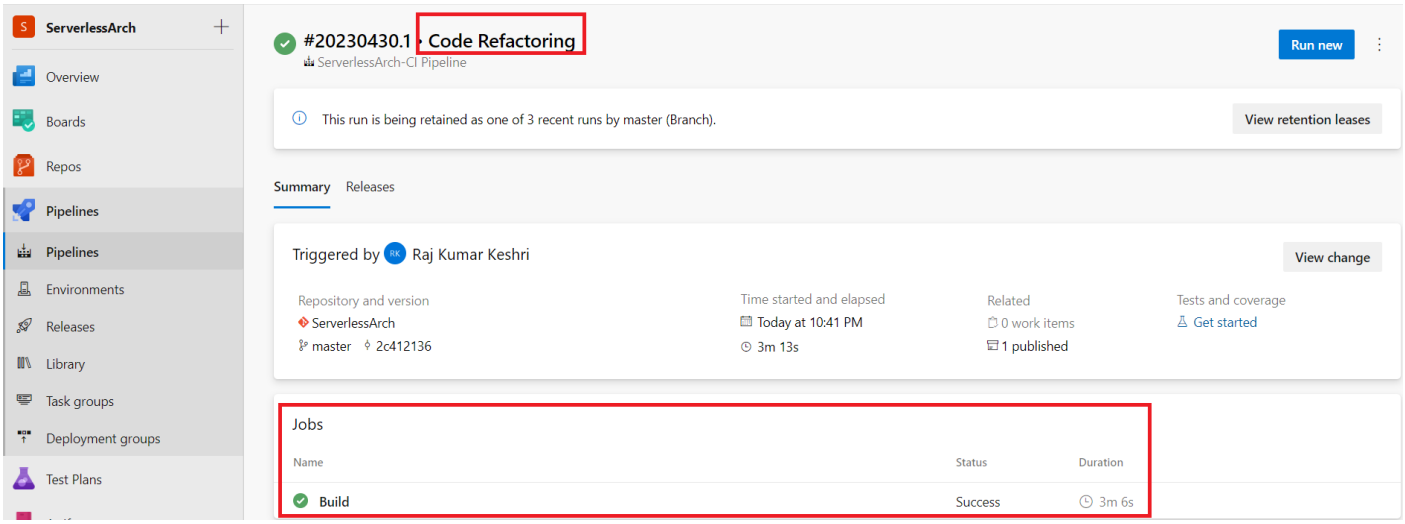


Figure 4:Completed CI pipeline

Conclusion

The accuracy percentage after about 100 epochs of using a dilated CNN to determine segment of an image is 78.36%. Increasing the epoch and trying with other loss function particularly logarithmic loss is keeping the accuracy at the same level and the loss at 0.7. This shows that the optimal performance for such a neural network architecture can achieved at a

faster pace [3]. For images that supports a color map and has limited color map representations this is quite a fast method to achieve the desired result. The dilation layer in the CNN increases the field of view and can help in identification of the segments and act as an agent of multi class segment classification.

Azure Functions is a serverless compute service that allows developers to quickly create, deploy, and manage applications without having to worry about underlying infrastructure. It is a powerful tool for building microservices and performing distributed computing tasks [14]. Azure Functions allows developers to focus on writing code and deploying applications without having to worry about provisioning, managing, or scaling the underlying infrastructure. In addition to its powerful compute capabilities, Azure Functions also provides a comprehensive set of tools for developing, deploying, and managing applications using a continuous integration and continuous delivery (CI/CD) pipeline. This pipeline enables developers to quickly and easily create, deploy, and manage applications without having to worry about the underlying infrastructure or complex configuration. The CI/CD pipeline allows developers to quickly build, test, and deploy applications, while also providing visibility into the entire development and deployment process. Using Azure Functions and a CI/CD pipeline, developers can quickly build and deploy applications in a fully automated fashion. This allows developers to focus on writing code, rather than spending time managing the underlying infrastructure. The CI/CD pipeline also provides a comprehensive set of tools for monitoring the performance and health of applications, as well as for updating and deploying new features. Overall, Azure Functions and a CI/CD pipeline provide a powerful and efficient way for developers to build, deploy, and manage applications in a serverless architecture. With Azure Functions, developers can quickly and easily create, deploy, and manage applications without having to worry about complex configuration or underlying infrastructure. Additionally, the CI/CD pipeline allows developers to quickly build, test, and deploy applications, while also providing visibility into the entire development and deployment process. By leveraging the power of Azure Functions and a CI/CD pipeline, developers can take advantage of the scalability and flexibility of serverless architecture.

Future Work

We want to extend this work by training and testing the same data set with the help of RCNN [1] with regional bounding spaces and using this with the combination of dilated CNN. The CNN part of the network is planned to contain the dilation layer. We expect this will increase the accuracy of the network further though the relationship with the training time must be investigated.

The use of Azure Functions along with CI/CD pipelines for serverless architectures is becoming increasingly popular due to its ability to reduce costs, improve scalability, and provide a more agile development environment. Azure Functions allow developers to deploy serverless applications in a short amount of time, and CI/CD pipelines help to automate the process of deploying and managing these applications. In the future, Azure Functions and CI/CD pipelines could be used to create complete end-to-end serverless architectures. This would involve the use of Azure Functions for the computation and storage of data, along with CI/CD pipelines for the automation of the deployment and management of the applications. The CI/CD pipeline would also provide the ability to quickly scale the application and add new features without manual intervention. Overall, the use of Azure Functions along with CI/CD pipelines for serverless architectures provides a number of benefits. It allows for faster development and deployment times, improved scalability, and cost savings. In the future, the use of these technologies could be extended to create end-to-end serverless architectures, machine learning models, and microservices architectures.

References

[1] Ramaswamy, S. (2021). DevOps on Azure, CI/CD Pipeline Automation with Azure Functions. Packt Publishing.

[2] Mittal, P. (2019). Cloud Native DevOps with Kubernetes: Building and Running Cloud Native Applications with Jenkins, Helm, and Kubernetes. Packt Publishing.

[3] Sivaraman, D. (2020). Cloud Native DevOps with Azure: Implementing DevOps Practices for Azure Applications. Packt Publishing.

[4] Berent, D. (2018). Continuous Delivery and DevOps: A Quickstart Guide. Packt Publishing.

[5] Chen, C., & Kong, Y. (2017). The DevOps 2.0 Toolkit: Automating the Continuous Deployment Pipeline with Containerized Microservices. Apress.

[6] Humble, J., & Farley, D. (2010). Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation. Addison-Wesley Professional.
[7] Li, Y. (2018). Performance analysis of serverless computing. ACM SIGAPP Applied Computing Review, 18(2), 2-17.
Sakr, S., Al-Fuqaha, A., & Guizani, M. (2017). Serverless computing: a survey. IEEE Communications Magazine, 55(7), 92-98.
[8] Souza, R. C. S., Herrmann, E. L., & Oliveira, J. A. (2020). An overview of serverless computing: Benefits
[9] Bull, J. (2016). Continuous Integration and Deployment with Docker and Jenkins. Packt Publishing.
[10] Microsoft. (2020). Azure Functions Documentation. Retrieved from https://docs.microsoft.com/en-us/azure/azure-functions
[11] Microsoft. (2020). Visual Studio Team Services Documentation. Retrieved from https://docs.microsoft.com/en-us/azure/devops/user-guide/index?view=azure-devops
[12] Microsoft. (2020). Continuous Delivery with Azure Functions and Visual Studio Team Services. Retrieved from https://learn.microsoft.com/en-us/azure/azure-functions/functions-how-to-azure-devops
[13] Niederstrasser, N., Hochstein, L., & Bakr, H. (2017). Automated deployment via DevOps: A systematic literature review. International Conference on Evaluation of Novel Approaches to Software Engineering, (pp. 67-78). Springer, Cham.
[14] Shaw, Ankit Kumar, et al. "Scalable IoT Solution using Cloud Services–An Automobile Industry Use Case." 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC). IEEE, 2020.
[15] Mohapatra, Debaniranjan, Amit Chakraborty, and Ankit Kumar Shaw. "Exploring novel techniques to detect aberration from metal surfaces in automobile industries." Proceedings of International Conference on Communication, Circuits, and Systems: IC3S 2020. Springer Singapore, 2021.
[16] Chakraborty, Amit, Ankit Kumar Shaw, and Sucharita Samanta. "On a Reference Architecture to Build Deep-Q Learning-Based Intelligent IoT Edge Solutions." Convergence of Deep Learning In Cyber-IoT Systems and Security (2022): 123-146.