SZAKDOLGOZAT

Nyilas Péter
Mérnökinformatikus hallgató részére

Közlekedési objektumok detekcióját megvalósító neurális hálós modell tanítása és magyarázata

A közlekedési objektumok, mint például autók, gyalogosok, kamionok, motorkerékpárok, biciklik és egyéb dinamikus közlekedési szereplők felismerése kritikus fontosságú feladat az önvezető, illetve a vezetést támogató rendszerek, valamint közlekedési monitoring rendszerek fejlesztése során. A neurális hálókon alapuló objektumdetekciós módszerek lehetővé teszik, hogy ezek az objektumok valós időben és nagy pontossággal detektálhatók legyenek.

A szakdolgozat célja egy olyan mély neurális hálózat kialakítása és tanítása, amely képes a közlekedési objektumok automatikus felismerésére és azonosítására képek alapján. Emellett az alkalmazás biztonságkritikus jellege miatt a hallgatónak meg kell vizsgálnia a modell magyarázhatósági aspektusait is, különös tekintettel a modell interpretálhatóságára.

A hallgató feladatának a következőkre kell kiterjednie:

- Végezzen irodalomkutatást a konvolúciós mély neurális hálók és a magyarázó technikák témájában!
- Végezzen adatgyűjtést, és készítse elő az adatokat a háló tanításához és validálásához!
- Végezze el a választott objektumdetekcióra használható neurális háló tanítását és értékelje ki annak teljesítményét!
- Alkalmazza a megismert modelfüggő és modelfüggetlen magyarázó módszereket a modell döntéseinek elemzésére!
- Alkalmazzon legalább két magyarázó módszert, és hasonlítsa össze azok hatékonyságát és eredményeit!

Tanszéki konzulens: Dr. Hullám Gábor, docens

Budapest, 2024.09.26.

……..
Dr. Dabóczi Tamás
tanszékvezető, egyetemi tanár

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Mesterséges Intelligencia és Rendszertervezés Tanszék

1117 Budapest, Magyar Tudósok krt. 2. I. ép. I.E.444.
Telefon: 463-2057, Fax: 463-4112
http://www.mit.bme.hu • e-mail: mitadm@mit.bme.hu

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Artificial Intelligence and Systems Engineering

Training and Interpretation of a Neural Network Model for Traffic Object Detection

Bachelor's Thesis

Author
Péter Nyilas

Advisor
dr. Gábor Hullám

October 13, 2024

Contents

HALLGATÓI NYILATKOZAT

Alulírott *Nyilas Péter*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2024. október 13.

Nyilas Péter
hallgató

Kivonat

Jelen dokumentum egy diplomaterv sablon, amely formai keretet ad a BME Villamosmérnöki és Informatikai Karán végző hallgatók által elkészítendő szakdolgozatnak és diplomatervnek. A sablon használata opcionális. Ez a sablon LaTeX alapú, a *TeXLive* TeX-implementációval és a PDF-LaTeX fordítóval működőképes.

Abstract

This document is a LaTeX-based skeleton for BSc/MSc theses of students at the Electrical Engineering and Informatics Faculty, Budapest University of Technology and Economics. The usage of this skeleton is optional. It has been tested with the *TeXLive* TeX implementation, and it requires the PDF-LaTeX compiler.

1 Introduction and Motivations

The accurate detection of objects that are relevant to the movement of vehicles is a critical task in the field of automotive and transportation systems. Numerous scenarios can arise where different types of object detection is required. The development of a highly sophisticated automated driving system requires the efficient gathering and processing of data from sensors located in the chassis to create a virtual model of the surrounding world.

The primary challenge of developing autonomous vehicles is the development of systems to create a reliable and accurate perception of the vehicle's environment. One of the most widely used solutions is the use of camera sensors as they are able to provide rich visual data that can be processed rapidly and can give the most information by itself. The goal of this project was to develop a software capable of fully fulfilling this need.

Nevertheless, the most significant limitation of these models is that they are frequently regarded as black boxes, lacking a straightforward correlation between input and output. This lack of transparency in how models arrive at conclusions has led to the development of numerous model interpretation techniques, as evidenced by the work of a number of different researchers [2]. These techniques vary in their approach, but they are all aiming to facilitate an insight into the decision-making process of the model.

The complexity of interpreting models in these domains arises from the intricate nature of the data and the sheer size of models used. This complexity makes it challenging to trace the decision-making process, yet it is essential for identifying potential biases, improving model performance, and gaining the trust of users and regulatory bodies. Therefore, with the will of enhancing the interpretability of this project's model is vital for maintaining safety and ethical standards, given the nature of the application.

The secondary goal of this project was to try and implement different interpretation methods, aimed to interpret the model and try to create explanations locally about the output our model produces.

1.1 Structure

The main parts of my thesis are as follows:

- Literature study about Image processing Deep Convolutional Neural Networks
- Yolo architecture and general overview on training and infrastructure.
- Cityscapes Dataset and data processing and representation.
- Evaluation of model performance
- Literature review about model-agnostic and model-dependent interpretation methods
- Implementation, use and evaluation of EigenCAM, LIME and SHAP.
- To analyze the results and draw conclusions on the effectiveness of the model and the XAI solutions for traffic object detection.

2 Image Processing Deep Neural Networks

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep learning models that are mostly used for computer vision tasks such as image classification, object detection, and segmentation. Given their ability to learn spatial hierarchies of features and their various types of data representations, they are perfect for the task of traffic object detection. Many recent advancements in computer vision have been made possible by CNNs, including the development of models like YOLO, Faster R-CNN, and ResNet. A dozen of these models have been developed, each with its own unique architecture and capabilities, but all of them share the same basic principles discussed by [1]

The name of the class come from their use of convolutional layers, which apply filters to input data to extract features, outside convolutional layers, CNNs also contain pooling layers, fully connected layers, and activation functions.

Convolutional layers Convolutional layers are the cornerstones building blocks of convolutional neural networks (CNNs). Convolution operations are applied to the input data using filters (or kernels) that is processing the input image. The process enables the model to identify local patterns, including edges, textures, and shapes. Each convolutional layer generates a set of feature maps, which indicate the presence of diverse features in the input. During the training phase, the parameters of these filters are learned, allowing the network to enhance feature extraction for specific tasks.

Pooling layers The objective of pooling layers is to reduce the spatial dimensions of feature maps, thereby decreasing the number of parameters and computations in the network. Furthermore, this process enhances the model's resilience to minor translations in the input data by mitigating the effects of potential outliers. The most prevalent forms of pooling are max pooling and average pooling. Max pooling selects the maximum value from a specified window, whereas average pooling computes the average. By down-sampling the feature maps, pooling layers assist in maintaining the most important features while discarding less critical information from the picture.

Fully connected layers Fully connected layers, also referred to as dense layers, are typically situated at the conclusion of convolutional neural network (CNN) architectures. In these layers, each neuron is connected to every neuron in the preceding layer. This structure enables the model to integrate information from all features and make final predictions. Fully connected layers are particularly crucial for classification tasks, where they compute the output probabilities for each class based on the features extracted by the preceding layers. Regularisation techniques, such as dropout, are frequently employed in these layers to prevent overfitting.

Sampling layers Sampling layers are used to reduce the dimensionality of the data while ensuring the preservation of the input's essential features. Sampling can entail techniques such as sub-sampling or strided convolutions, whereby a specific stride is applied to the convolution operation in order to down-sample the feature maps.

Activation Functions are used in CNNs to introduce non-linearity into the model. They help the model learn complex patterns and make accurate predictions. The most commonly used activation functions are ReLU, Sigmoid, and Tanh. In light of the work of Siddharth Sharma, Simone Sharma and Anidhya Athaiya (see at [4]), it is evident that there are other types of activation functions such as BSF and ELU that could be employed. However, as these are not utilised by the examined model(Yolov8), a detailed discussion of them will not be provided.

ReLU The Rectified Linear Unit (ReLU) is one of the most widely used activation functions in convolutional neural networks (CNNs). It is defined as $f(x) = \max(0, x)$ by [4]. This function introduces non-linearity while maintaining a simple and efficient computation. ReLU helps mitigate the vanishing gradient problem, allowing models to learn faster and perform better. However, it can suffer from the "dying ReLU" problem, where neurons become inactive and only output zeros, particularly during the training of deep networks.

Sigmoid The Sigmoid function maps input values to a range between 0 and 1, making it useful for binary classification problems. It is defined as $f(x) = \frac{1}{1+e^{-x}}$ by [4]. While the Sigmoid function provides smooth gradients, it is prone to the vanishing gradient problem, especially for large positive or negative input values. This can slow down the training of deep networks, which is why it is often replaced by other activation functions in hidden layers, though it still finds use in the output layer for binary classification tasks.

Tanh The Hyperbolic Tangent (Tanh) function is another activation function that maps input values to a range between -1 and 1. It is defined as $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ by [4]. Tanh is zero-centered and centrically symmetrical, which helps in centering the data and can lead to faster convergence during training. However, it still suffers from the vanishing gradient problem for large input values, though to a lesser extent than the Sigmoid function.

3 The Model, its training and data management

3.1 Introduction to YOLO

The YOLO (You Only Look Once) model is a cutting-edge object detection system that has gained a reputation for its speed and accuracy. It is based on the YOLO algorithm , which is a real-time object detection algorithm developed by Joseph Redmon and Ali Farhadi in 2015[**?**].

Unlike traditional object detection methods that apply classifiers to various sections of an image, YOLO approaches the problem as a single regression problem. It divides the image into a grid and simultaneously predicts bounding boxes and class probabilities for each grid cell, allowing it to detect multiple objects in a single pass. This architecture not only enhances speed but also improves detection accuracy by reducing the number of false positives.

The YOLO model has evolved through multiple versions, with improvements in both performance and varying capability. Subsequent versions of the model, including YOLOv2, YOLOv3, and the latest YOLOv5 and YOLOv7, as well as Yolov8 (with Yolov10 and 11 forthcoming), have introduced advancements in network architecture, feature extraction,

and training techniques. These developments have rendered YOLO a suitable candidate for a plethora of applications, including autonomous vehicles, as evidenced by the author's own observations, surveillance systems, and real-time video analysis.

A further noteworthy attribute of the Yolo-type neural networks is their scalability and versatility in terms of model architecture. These models are available in a range of sizes (*nano, small, medium, large, extralarge*) and with a variety of detection types (*semantic segmentation, bounding boxes, oriented bounding boxes, instance segmentation*), which can be deployed in diverse scenarios.

For my work, I chose the **Yolov8m** configuration, which stands for **Yolo version 8 medium**. My choice was made on the bases of previous experience with this model architecture and the popularity of its applications.

3.2 Model architecture

This network, like many others in the CNN family, has a distinctive architectural configuration that enables the execution of intricate tasks such as object detection and classification. The system is constituted of three distinct and discrete components, each comprising a unique set of layers that perform specific and separate functions.

Backbone The YOLOv8 model is based on convolutional neural networks (CNNs) that have been specifically designed to capture essential features from input images. It comprises multiple layers of convolutional operations (called Conv and a complex layer called C2F) that extract progressively more sophisticated representations. This structure emphasises both depth and computational efficiency, enabling the model to discern subtle details while maintaining rapid processing capabilities. Innovations such as skip connections and normalization techniques are employed to enhance the learning dynamics and improve the model's robustness to variations in input conditions.

Neck In the YOLOv8 architectural design, the neck serves as an intermediary between the backbone and the head. The primary function of the neck is to consolidate features from the various levels of the backbone, thereby enhancing the model's ability to detect objects across different scales. By employing strategies such as feature fusion or pyramid pooling, the neck effectively integrates both coarse and fine features, enabling the model to better handle overlapping objects and diverse scene contexts. This feature aggregation is crucial for optimising the detection performance, as it allows the model to harness a comprehensive range of information.

Head The head of the YOLOv8 model is responsible for generating the final outputs, which are based on the features that have been processed through the neck. The final stage of the YOLOv8 model translates the aggregated feature maps into bounding box predictions and associated class scores for each detected object. This section typically employs a combination of convolutional and fully connected layers to refine these outputs, ensuring they are accurate and meaningful. Additionally, the head may implement techniques such as adaptive anchors or confidence scoring to enhance the localisation and classification accuracy. By effectively synthesising the rich feature information, the head enables YOLOv8 to achieve high-performance object detection suitable for a wide array of applications.

3.3 Training

The training process [**?**] for the YOLOv8 model involves feeding its input with a large dataset of labeled images. The model learns to identify objects by minimizing the difference between its predictions and the actual labels trough multiple iterations: After a training cycle ,if the default settings are kept, a validation function (so called val) is run, to determine more information on the model's performance on pictures that are not present in the training set. The training process is iterative, with the model adjusting its parameters to improve its predictions over time. This process is computationally intensive and requires access to powerful hardware, such as GPUs.

The training process involves several key steps, including data preprocessing, model initialization, loss calculation, and parameter optimization. The model is trained using a technique called backpropagation,which involves adjusting the model's weights based on the error between its predictions and the ground truth labels. The Yolov8 model uses a number of different loss functions to measure the difference between its predictions and the actual labels:

- CIoU(Complete Intersection over Union) loss for bounding box regression to improve localisation accuracy.
- DFL loss (Distribution Focal Loss) It helps the model to more accurate classification.
- VFL loss (Varifocal Loss) It's designed to address imbalances and uncertainties in classification tasks.

I opted for training the model on my local machine, using a single GPU, while it provided me with sufficient performance , the training time was significantly longer than it would have been on a more powerful cloud machine. To reduce the chance of overfitting, the training dataset is typically divided into training and validation sets, with the latter used to evaluate the model's performance on unseen data. The trainings batch size where set to 3, which is not common, but it was necessary to fit the model on the GPU, to optimise training time.

3.4 Training and evaluation with the help of MLOps solutions

In order to monitor the performance of the model and to facilitate the visualisation of the learning process, as well as to organise the experiments, an online machine learning monitoring solution, Comet.ml, has been selected. This tool is capable of tracking the training process in real time and of visualising the properties of the model on a user-friendly dashboard, which can be accessed from any device with an internet connection. The Comet.ml platform also provides a range of features that can be used to compare different experiments, such as hyperparameter tuning, model versioning, and collaboration tools, while also offering a comprehensive set of APIs for integration with other tools and platforms.

I used it to monitor the training process of the YOLOv8 model and to evaluate its performance on the Cityscapes dataset, as well as to compare the results with other versions of the model configurations.

3.5 Dataset and formats

Cityscapes The Cityscapes dataset is a large-scale dataset [**?**] used for training and evaluating object detection models. It contains high-resolution images of urban scenes, with detailed annotations for various objects such as cars, pedestrians, and traffic signs. The dataset is widely used in the field of computer vision for tasks such as semantic segmentation and object detection.

I chose the gtFine dataset, consisting precisely labelled segmentation masks. Based on their work [**?**] the set consists of 5000 images, with a resolution of 1024x2048 pixels, and annotations for 30 classes of objects. It is divided into three subsets: training, validation, and test, with 2975, 500, and 1525 images, respectively. The dataset is annotated using pixel-level segmentation masks, which provide detailed information about the location and shape of objects in the scene. However for the purpose of this work, the annotations were converted into bounding box format, which is more suitable for object detection tasks.

Format conversion and datatypes The images and annotations are converted into a format that the YOLOv8 model can process, which is a text file containing the image path and the coordinates of the bounding boxes for each object in the image. This process uses a custom script that reads the annotations from the Cityscapes dataset and converts the labels into labels whose classes are filtered and transformed into the grouping I chose for this project. The classes were grouped into five categories:

- small vehicle*(usually cars, which are for personal use)*,
- large vehicle*(busses, trucks and other large non personal vehicles)*,
- two wheelers*(bicycles nad motorcycles)*,
- On-rails*(trains and trams though the smaller Fine dataset didn't include any)*
- and person *(pedestrian, and rider)*.

The script also converts the semantic segmentation masks into bounding boxes, trough finding the most extreme points and create a bounding box around them. This converted output is then saved in a text file, which is used as input for the YOLOv8 model.

At the end the structure of the data is as follows:

- Root(gtFine): The root directory of the Cityscapes dataset, which contains the images and annotations.
  - labels: Folder containing the images in the dataset, broken down into training, validation, and test sets and further divided into subfolders based on the city where the images were captured.
  - labels: The class label for each object in the image.
  - train.txt: The training set, which contains the paths to the training images and their corresponding annotations.
  - val.txt: The validation set, which contains the paths to the validation images and their corresponding annotations.
  - test.txt: The test set, which contains the paths to the test images and their corresponding annotations.

- Descriptors: The folder containing the class labels and their corresponding indices, as well as path set descriptor txt-s. It's used by the model to determine the classes, their indices and the paths to the images.

3.6 Model evaluation

The evaluation of the YOLOv8 model is performed using a set of metrics that measure its performance on the Cityscapes dataset. These metrics include precision, recall,mAP, and IoU, which are commonly used in object detection tasks.

4 Model Interpretation

Model interpretation is a critical aspect of machine learning that aims to explain the decision-making process of models. Interpretable models are essential for building trust with users, identifying biases, and improving model performance. In this section, I discuss the importance of model interpretation and review various interpretation methods. In their work [2], Liang et al. highlight the significance of model interpretation techniques in providing insights into the decision-making process of models. They sort these techniques into two categories:

- Data Driven or more commonly Model-agnostic methods
- Model Driven or Model-specific methods

4.1 Importance of Interpretation

Interpreting machine learning models is essential for understanding their decision-making processes. It helps in identifying biases, improving model performance, and building trust with users. Interpretation techniques provide insights into how models make predictions and highlight the most influential features.

4.2 Model Agnostic Methods

Based on the work of [2], model-agnostic interpretation methods can be divided into two categories:

- Perturbation-based interpretation
- Game theoretic approach

As discussed in [2], these are designed to explain model predictions without relying on the internal structure of the model, making them applicable to a wide range of models. For image processing purposes, these methods are particularly easy to understand and implement. Both the game-theoretic and perturbation-based interpretation methods involve altering the input data and observing the resulting changes in the model's predictions on the modified image.

Perturbation-based Interpretation Perturbation-based interpretation methods are model-agnostic techniques that explain model predictions by perturbing the input data. These methods generate perturbed samples by adding noise to the input image and observe the changes in the model's predictions. This masking is the main principle behind these methods. The most common type of masking is occlusion, where parts of the image are covered to determine their importance for the model's output. It can be interpreted as a form of feature selection, where the model's output is evaluated based on the presence or absence of specific features.

It works in a similar way to the human visual system and the way we perceive the visual world, so that by obscuring different parts of an object, we can significantly alter our own eye's perception of the object. By analysing the effect of perturbations on the model's output, these methods identify the most important features for a given prediction.

Game theoretic approach with perturbation Game-theoretic approaches with perturbation are model-agnostic interpretation methods that leverage cooperative game theory to explain model predictions. This translates to decomposition of the input image into a number identical, equally-sized components, referred to as "features". The aforementioned parts constitute a set. In accordance with the implementation of this approach, SHAP, assigns a Shapley value to each feature. Based on its contribution to the model's prediction. This is calculated by considering all potential combinations of parts. These combinations represent all the subsets of the set of parts, and the Shapley value is calculated for each subset. The greater the variation in the model's prediction when a feature is added or removed, the higher the Shapley value. By considering all the potential combinations of features, these methods offer consistent and accurate explanations for model predictions.

Comparison of Model Agnostic Methods Based on the work of [2], these model agnostic interpretation methods have their strengths and weaknesses. Two of the most popular model-agnostic interpretation methods are LIME(as an example for Perturbation-based Interpretation) and SHAP(as an example for Game theoretic approach with perturbation). Through their example I will illustrate the differences between the two methods.

LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) while using different approaches, there are both using some form of perturbation to explain the model's predictions. LIME approximates the model locally by fitting a simple interpretable model around the prediction of interest, providing local explanations by perturbing the input data and observing the changes in the model's predictions. It is generally faster and simpler, making it suitable for quick, local explanations.

On the other hand, SHAP is based on cooperative game theory and uses Shapley values to attribute the contribution of each feature to the prediction. It provides both local and global explanations by considering all possible combinations of features, producing consistent and theoretically sound explanations.

Based on the discussions by [**?**], Kernel SHAP, a variant of SHAP, is particularly useful for image processing tasks, as it can handle high-dimensional data efficiently. It is based on the idea of approximating the model with LIME and using the Shapley values to explain the model's predictions. This approach combines the strengths of both LIME and SHAP, providing accurate and efficient explanations for image processing models.

However, SHAP is more computationally intensive due to the need to consider all possible feature combinations. During the interpretation scripts run, SHAP had 4 times larger

GPU memory usage than LIME, while the runtime was 100 times longer. While LIME is flexible and can be applied to any model and data type, SHAP offers a more comprehensive and theoretically grounded approach at a higher computational cost.

4.3 Model Specific Methods

The base idea behind this category is to ground our interpretation on the models internal state. By this in the field of image processing, it aims to somehow visualize the inner state of the model, and project it back to the input image. Multiple methods exist in this category, such as Class Activation Maps, Gradient-based methods. In the following paragraphs, I will discuss the Class Activation Maps based on the work of [**?**] and its implmentation in the next section. Furthermore, I will discuss the Gradient-based methods based on the work of [**?**].

Class Activation Maps

Gradient-based Methods

4.4 Comparison of Interpretation Methods

5 Model interpretation using external solutions

5.1 Application of Model agnostic methods

Local Interpretable Model-agnostic Explanations LIME is a popular model-agnostic interpretation method that explains individual predictions by approximating the model locally with an interpretable model. It perturbs the input data and observes the changes in the model's predictions to identify the most important features.

Shapley Additive explanations SHAP is another model-agnostic method that provides consistent and accurate explanations for model predictions. It is based on cooperative game theory and assigns a Shapley value to each feature, representing its contribution to the prediction.

5.2 Application of Model specific methods

EigenCAM EigenCAM is a model-specific interpretation method that visualizes the regions of an image that are most important for a model's prediction. It computes the principal components of the feature maps and highlights the areas that contribute the most to the final decision.

Summary

This document presents a comprehensive study on the training and interpretation of a neural network model for traffic object detection. The key points discussed in the chapters are summarized as follows:

- **Model, Training, and Data**: The YOLOv8 model architecture, including its backbone, neck, and head components, is detailed. The training process and the Cityscapes dataset used for training are also discussed.
- **Model Interpretation Using External Solutions**: The importance of model interpretation is highlighted. Various model-agnostic methods such as LIME and SHAP, as well as model-specific methods like EigenCAM and EigenGradCAM, are explained.

The study concludes that effective training and interpretation of neural network models are crucial for accurate and reliable traffic object detection.

Bibliography

[1] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018. ISSN 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2017.10.013. URL https://www.sciencedirect.com/science/article/pii/S0031320317304120.

[2] Yu Liang, Siguang Li, Chungang Yan, Maozhen Li, and Changjun Jiang. Explaining the black-box model: A survey of local interpretation methods for deep neural networks. *Neurocomputing*, 419:168–182, 2021. ISSN 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.08.011. URL https://www.sciencedirect.com/science/article/pii/S0925231220312716.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. 2015. URL http://arxiv.org/abs/1506.02640.

[4] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.