|  |  |  |  |
| --- | --- | --- | --- |
|  | Politechnika Bydgoska im. J. J. Śniadeckich  Wydział Telekomunikacji, Informatyki i Elektrotechniki  **Zakład Systemów Teleinformatycznych** | |  |
| **Przedmiot** | Wdrażanie Technik Uczenia Maszynowego | | |
| **Prowadzący** | mgr inż. Martyna Tarczewska | | |
| **Temat** | Klasyfikacja Obrazów – Pizza vs Niepizza | | |
| **Student** | Zuzanna Tarazewicz | **Nr Indeksu** | 116954 |
| **Ocena** |  | **Data Oddania** |  |

Spis Treści

[Specyfikacja Projektu 3](#_Toc170825851)

[Zbiór Danych 3](#_Toc170825852)

[Pobranie Danych 3](#_Toc170825853)

[Podział danych 3](#_Toc170825854)

[Eksploracyjna Analiza Danych 4](#_Toc170825855)

[Tworzenie Modelu 4](#_Toc170825856)

[Splotowe Sieci Neuronowe 4](#_Toc170825857)

[Sieć Pretrenowana VGG16 4](#_Toc170825858)

[Sieć Pretrenowana Resnet50 4](#_Toc170825859)

[Porównanie jakości modeli 4](#_Toc170825860)

[Tworzenie Frontendu aplikacji 4](#_Toc170825861)

[Uruchomienie programu przez Docker 4](#_Toc170825862)

[CI/CD 4](#_Toc170825863)

[Chmura 5](#_Toc170825864)

[Wnioski 5](#_Toc170825865)

# Specyfikacja Projektu

* Dane: <https://www.kaggle.com/datasets/carlosrunner/pizza-not-pizza>
* Repozytorium:
* Temat: Klasyfikacja Obrazów: Pizza vs Niepizza
* Chmura: Amazon Web Services

Celem projektu jest stworzenie modelu uczenia maszynowego klasyfikującego obrazy. Dany model będzie rozpoznawał czy na obrazie będzie znajdować się pizza, czy nie. Zbiór danych składa się z 787 obrazów pizzy w zbiorze treningowym, 98 obrazów pizzy w zbiorze walidacyjnym, 99 obrazów pizzy w zbiorze testowym, 786 zdjęć na których nie znajduje się pizza w zbiorze treningowym, 98 obrazów na których nie znajduje się pizza w zbiorze walidacyjnym I 99 obrazów na których nie znajduje się pizza w zbiorze testowym.

Kolejnym celem projektu jest stworzenie CI/CD projektu i wyszkolenie modelu na chmurze wykorzystując techniki MLOps.

Do wykonania projektu zostały wykorzystane:

* Python 3
* Amazon SageMakerLab
* Docker
* GitHub

# Zbiór Danych

## Pobranie Danych

W celu pobrania danych z kaggle.com wykorzystana została biblioteka opendatasets w której znajduje się funkcja download, która pobiera dane znajdujące się pod podanym linkiem. Po uruchomieniu programu należy podać login konta na kaggle i specjalnie wygenerowany klucz, który znajduje się w pliku .json po pobraniu.

Wygląda to w następujący sposób:

A black screen with red text

Description automatically generated

downloadData.py

from opendatasets import download

link = 'https://www.kaggle.com/datasets/carlosrunner/pizza-not-pizza'

download(link)

print('Download complete!')

## Podział danych

Zbiór znajdujący się na kaggle nie dzieli się na konkretne zbiory, a klasy. W celu podziału na zbiór treningowy, walidacyjny i testowy wykorzystano bibliotekę split-folder. W argumentach funkcji ratio pobierane są m.in ścieżka do zbioru, który ma zostać podzielony, ścieżka w której ma zostać zapisany i w jakich proporcjach mają zostać utworzone foldery. Jeśli ratio ma długość krotki równą dwa, to zbiór dzieli się na zbiór treningowy i testowy, a jeśli ma długość krotki równą trzy – to zbiór dzieli się na zbiór treningowy, testowy i walidacyjny.

split-folders.py

import splitfolders as s

s.ratio('pizza-not-pizza/pizza\_not\_pizza', output='pizza-notpizza', seed=1337, ratio=(.8, .1, .1), group\_prefix=None, move=True)

# Eksploracyjna Analiza Danych

Następnym krokiem w projekcie było stworzenie szybkiej analizy danych. Dzięki niej można zaplanować jakie podejście i metryki należy stosować itp.

W celu obliczenia ilości danych została stworzona funkcja fileCounter pobierająca ścieżkę do danego folderu. Wykorzystuje ona bibliotekę os. Została stworzona aby nie musieć powtarzać tego kodu sześciokrotnie.

def fileCounter*(directory: str)*:

lst = os.listdir(directory)

fileCount = len(lst)

return fileCount

Dzięki niej przy użyciu biblioteki matplotlib udało się wykonać wykres słupkowy (który jest dobrą i przejrzystą reprezentacją danych tego typu).

W liście classes znajdują się nazwy klas, a w słowniku classesCount znajdują się konkretne ilości każdego podzbioru, które występują w formie dwuelementowej listy. Za pomocą pętli for można utworzyć przejrzysty wykres słupkowy, który prezentuje dystrybucje danych w poszczególnych klasach i poszczególnych zbiorach.

A graph with numbers and a bar

Description automatically generated with medium confidence

Kod tworzący ten wykres:

classes = ['Pizza', 'Not Pizza']

classesCount = {

'Training Data': training,

'Validation Data': validation,

'Testing Data': testing

}

x = np.arange(len(classes))

width = 0.33

multiplier = 0

fig, ax = plt.subplots(layout='constrained')

for dataset, eachClassCount in classesCount.items():

offset = width \* multiplier

rects = ax.bar(x + offset, eachClassCount, width, label = dataset)

ax.bar\_label(rects, padding=3)

multiplier += 1

ax.set\_ylabel('Quantity')

ax.set\_ylabel('Class')

ax.set\_title('Pizza or Not Pizza by Dataset')

ax.set\_xticks(x + width, classes)

ax.legend(loc = 'center', ncols = 3)

ax.set\_ylim(0, 900)

plt.show()

# Tworzenie Modelu

Modele zostały wytrenowane za pomocą Amazon SageMaker Lab ze względu na szybkość i cenę rozwiązania (4h darmowego uczenia na GPU).

Wszystkie sieci zostały zaprogramowane za pomocą biblioteki PyTorch.

## Cechy wspólne

### Hiperparametry

* Rozmiar wsadu: 64
* Stała ucząca: 2,5 x 103 (0,0025)
* Weight Decay: 1 x 10-5
* Ilość kanałów 3
* Rozmiar obrazu: 224 x 224 px

### Augmentacje Obrazu

* Zmiana rozmiaru
* Losowe przerzucenie horyzontalne
* Losowe przerzucenie Wertykalne
* Losowe obrócenie obrazu
* Zmiana jasności, kontrastu i saturacji

## Splotowe Sieci Neuronowe

Klasa sieci:

class CNN(nn.Module):

def \_\_init\_\_*(self)*:

super(CNN, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(3, 32, kernel\_size=3, stride=1, padding=1)

self.conv2 = nn.Conv2d(32, 64, kernel\_size=3, stride=1, padding=1)

self.conv3 = nn.Conv2d(64, 128, kernel\_size=3, stride=1, padding=1)

self.fc1 = nn.Linear(128 \* 28 \* 28, 256)

self.fc2 = nn.Linear(256, 128)

self.fc3 = nn.Linear(128, 2)

self.dropout = nn.Dropout(0.6)

self.pool = nn.MaxPool2d(kernel\_size=2, stride=2)

def forward*(self, x)*:

x = self.pool(nn.functional.leaky\_relu(self.conv1(x)))

x = self.pool(nn.functional.leaky\_relu(self.conv2(x)))

x = self.pool(nn.functional.leaky\_relu(self.conv3(x)))

x = x.view(x.size(0), -1)

x = nn.functional.leaky\_relu(self.fc1(x))

x = self.dropout(x)

x = nn.functional.leaky\_relu(self.fc2(x))

x = self.dropout(x)

x = self.fc3(x)

return x

## Sieć Pretrenowana VGG16

## Sieć Pretrenowana Resnet50

## Porównanie jakości modeli

# Tworzenie Frontendu aplikacji

# Uruchomienie programu przez Docker

A screenshot of a computer

Description automatically generated

# CI/CD

# Chmura

# Wnioski