Gender Classification Report

מטרה:

מטרת הפרוייקט היא ליצור מודל סיווג בינארי ולאמן אותו על מנת להגיע לאחוז דיוק מירבי של זיהוי תמונות פנים של גברים ונשים, ולסווגם בהתאם. המטרה בסוף הריצה היא לצאת עם מודל בעל אחוז גבוה של זיהוי פני גבר ואישה, נשים לב שזיהוי זה היא תכונה "אנושית" שכן מחשב אינו יודע להבדיל בין גברים ונשים, ולכן נשתמש ב-Machine Learning.

גישה:

כאשר מנתחים פעילויות אנושיות באמצעות למידת מכונה, זה יכול להיות שימושי להסיק מאפיינים כגון המין של האנשים המעורבים. אנחנו נתמקד בתת-בעיית ההכרה המגדרית, שנחקרה רבות בספרות, כאשר שתי בעיות עיקריות נותרו בלתי פתורות: כיצד לשפר את הדיוק בתמונות פנים אמיתיות, וכיצד להכליל את המודלים לביצועים טובים בהם מערכי נתונים חדשים. אנו מטפלים בבעיות אלו על ידי 200 אלף תמונות פנים וביצוע ניסויים שונים, תוך חקירת: ההבדל בביצועים בין רשתות הנוירונים (CNNs) בעומקים שונים וגישת מכונה וקטורית תומכת המשתמשת בתכונות דפוס בינארי מקומי על אותם נתוני אימון. ההשפעה של מידע משפיע על דיוק הסיווג.

ארכיטקטורה ומודל:

על מנת להגיע למטרה הנ"ל, נבנה ונאמן מודל של רשת נוירונים. רשת זו "תתאמן" על גבי ה-DataSet מ-Kaggle אותו התבקשנו להוריד, לאחר האימון על ה-Data Set, נרצה שהמודל יהיה אמין גם עבור תמונות שאינן לקוחות מה-Data Set, כלומר לא נרצה מצב של Overfitting.

תשתית הפרויקט בנויה ע"י ספריית pytorch אשר מספקת סביבת עבודה נוחה תחת תנאי הפרויקט.

ראשית נבצע הגדלה (augmentation) של הדאטה, בשלב זה נרצה להוסיף עוד תמונות בוורסיות שונות על מנת לשפר אחוז דיוק , ניקח את הדאטה שלנו ונעשה עליו מניפולציות. לדוגמא :נסבוב את התמונות, נשתמש בצבעי אפור למנוע רעשים ועוד..

נשתמש במודל "Sequential", מודל אידיאלי כאשר לכל שכבה ברשת הנוירונים יש קלט ופלט אחד בלבד. עבור ה-loss model נשתמש ב- "Cross Entropy Loss " שכן הסיווג שלנו הוא בינארי (שתי אופציות- גבר או אישה). ה-Optimizer בו בחרנו הוא gradient descent, Optimizer "שקול" יחסית שהתנודות בו אינן מהירות מדי לשני הכיוונים, אך יכול להגיע לאחוזי דיוק גבוהים עם המומנטום אליו נרצה להגיע. אך חשוב לזכור כי gradient descent מוצא מינימום מקומי ולא את הפתרון האידאלי.

לאחר אימון המודל נבצע בדיקת אחוז דיוק לקבוצת הוולידציה שלנו ולקבוצת המבחן .

-בקוד שלנו השתמשנו ברשת קונבולוציה (Convolutional Neural Network), שזוהי רשת נוירונים המשמשת לניתוח תמונה, הרשת בנויה מconvolution layers ע"י העברת ה-kernel ע"י הקלט.
נשים לב שהשכבות הן Fully-Connected ומכילות Convolution ו-Pooling, הראשון אחראי להעמקת הרשת וצמצום עלות החישוב של הרשת, הוא מקבל קלט רחב ומצמצם אותו למפה מופשטת יותר, ה-pooling משומש בתצורת max pooling, זוהי שיטת אגרגציה פופולרית המשמשת לחסכון בזמן עיבוד.

- כפי שציינו למעלה, בקוד שלנו אנחנו משתמשים במודל Sequential, כשמו כן הוא מודל רציף המשמש כאשר הקלט שלנו הוא זרמים של נתונים, במקרה שלנו תמונות.

- עבור ה-Optimizer שלנו השתמשנו ב-SGD (Stochastic gradient descent), Optimizer מאוד פופולרי, הוא בוחר באופן רנדומי נקודת דאטה בכל איטרציה ובכך מוריד את זמן החישוב באופן משמעותי מאוד.

סקירת הקוד:

Imported libraries:

```
!pip install skorch

from zipfile import ZipFile
import torch
torch.cuda.empty_cache()
torch.cuda.get_device_properties(0)
import os
import torchvision.datasets as datasets
import torchvision.transforms as transforms
from torchvision.models import resnet50
from torch import optim
import torch.nn as nn
from sklearn.metrics import accuracy_score
from skorch import NeuralNetClassifier
from skorch.helper import predefined_split
```

שלבי הקוד :

שלב 1:

תחילה נתחבר ל-Kaggle על מנת להוריד את ה-DataSet:

```
import os
os.environ['KAGGLE_USERNAME'] = "XXXX" #kaggle username
os.environ['KAGGLE_KEY'] = "XXXX" #kaggle key
!kaggle datasets download -d ashishjangra27/gender-
recognition-200k-images-celeba # api copied from Kaggle
```

יש לשים לב שעל המשתמש לשנות את שדה ה-Username וה-Key למשתמש שלו באתר.

לאחר מכן, נחלץ את קבצי ה-Data מה-Zip:

```
# Upload data and extract the contents
from zipfile import ZipFile
file_name = "/content/gender-recognition-200k-images-
celeba.zip"
```

```
with ZipFile(file_name, 'r') as zip:
  zip.extractall()
  print('done')
```

שלב 2:

נבצע Data Augmentation, נביא את את התמונות לאותו גודל, ו"נטען" אותן.

להלן הפונקציה האחראית לאוגמנטציה:

```
def load_data(data_folder, train):
    transform = {
        'train': transforms.Compose(
            [
                transforms.Resize([150, 150]),
                transforms.RandomCrop(130),
                transforms.ColorJitter(brightness = 0.5),
                transforms.RandomRotation(45),
                transforms.RandomHorizontalFlip(),
                transforms.ToTensor()]),
        'test': transforms.Compose(
            [transforms.ToPILImage(),
                transforms.Resize([130, 130]),
                transforms.ToTensor()]),
        'val':  transforms.Compose(
            [transforms.ToPILImage(),
                transforms.Resize([130, 130]),
                transforms.ToTensor()])
    }
    data = datasets.ImageFolder(
        root=data_folder, transform=transform['train' if train else 'test'])
    return data
```

טעינת הדאטה:

```
train_data = '/content/Dataset/Train'
test_data = '/content/Dataset/Test'
val_data = '/content/Dataset/Validation'

train_set = load_data(train_data, 'train')
val_set = load_data(val_data, 'val')
test_set = load_data(test_data, 'test')
```

שלב 3:

בניית המודל ותחילת האימון:

נשים לב, שבנינו רשת נוירונים רציפה בעלת 4 שכבות.

```
model = resnet50(pretrained=True)
model.fc = nn.Sequential(
                        nn.Linear(2048, 512),
                        nn.Dropout(0.5),
                        nn.Linear(512, 2),
                        nn.Dropout(0.5),
                        nn.Sigmoid()
                        )
lrscheduler = LRScheduler(policy='StepLR', step_size=7, gamma=0.1)
checkpoint = Checkpoint(f_params='best_model.pt', monitor='valid_acc_be
st')
progress = ProgressBar(batches_per_epoch=1000)
early_stop = EarlyStopping(monitor='valid_loss', patience=5)
optimizer = optim.SGD
loss_funcion = nn.CrossEntropyLoss
net = NeuralNetClassifier(
    model,
    optimizer=optimizer,
    optimizer__momentum=0.9,
    criterion=loss_funcion,
    max_epochs=10,
    train_split=predefined_split(val_set),
    iterator_train__shuffle=True,
    callbacks=[lrscheduler , progress , early_stop , checkpoint],
    #GPU
    device='cuda'

  #fit the model
  net.fit(train_set, y=None)
```

שלב 4 :

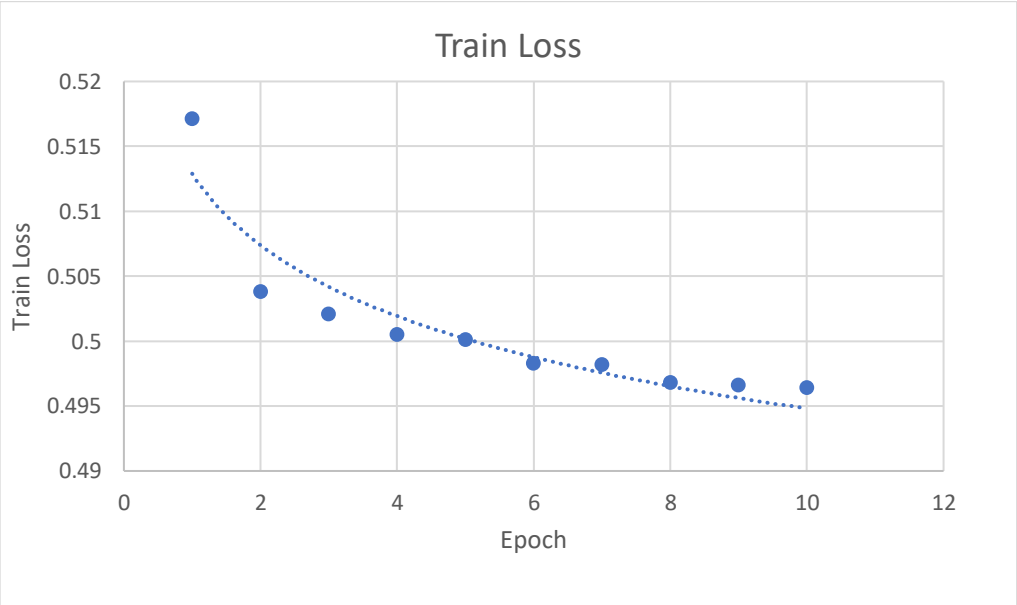
הדפסת התוצאות:

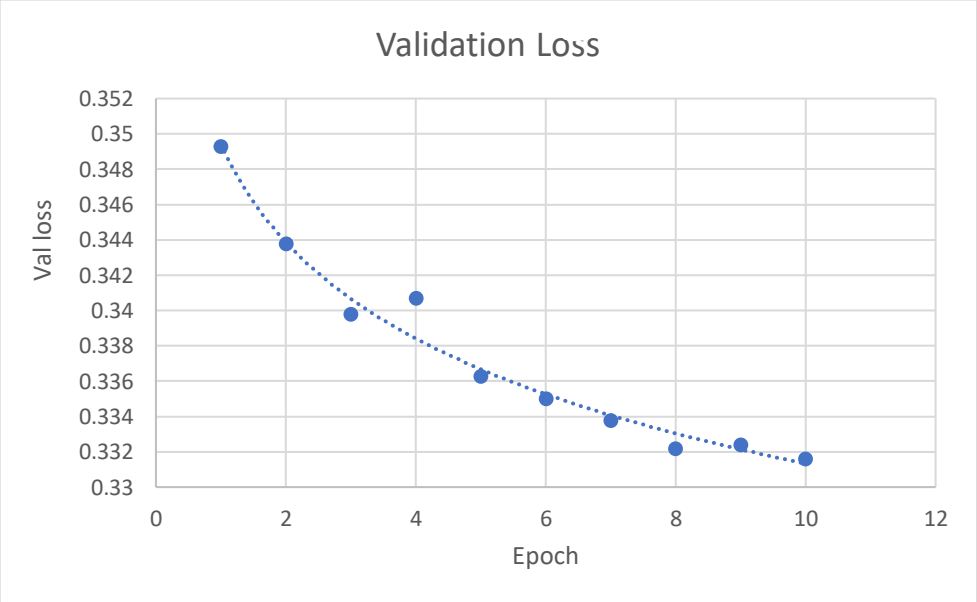
```
#acc val
y_pred = net.predict_proba(val_set)
```

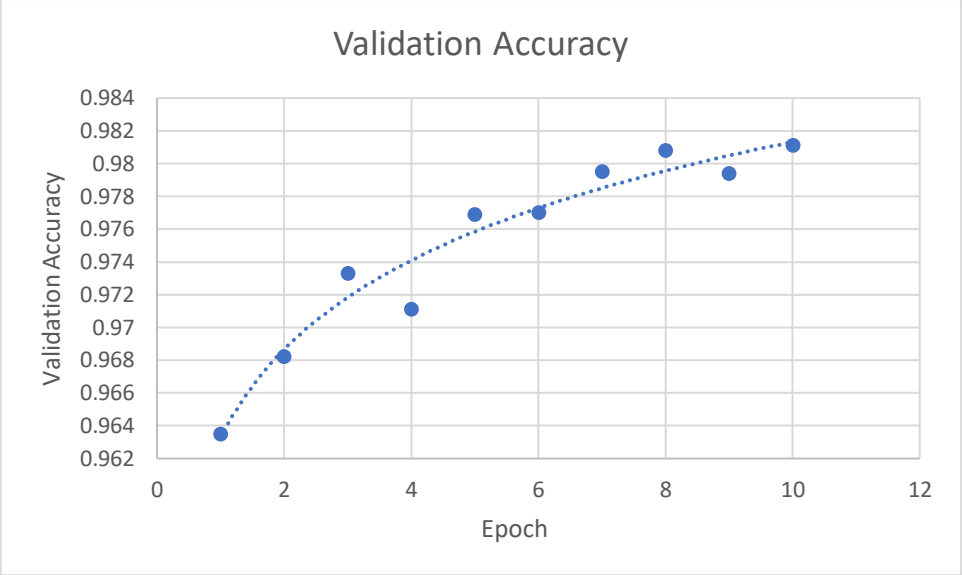
```
y_test = val_set.targets
acc_val = accuracy_score(y_test, y_pred.argmax(axis=1))
print('Validation Accuracy: ',acc_val)

#acc tests
y_pred = net.predict_proba(test_set)
y_test = test_set.targets
acc_test = accuracy_score(y_test, y_pred.argmax(axis=1))
print('Test Accuracy: ', acc_test)
```

תוצאות המודל:

במהלך ריצת הקוד הגענו לדיוק של 98.11% ול-loss של 33%, להלן הגרפים שלהם כתלות במספר שלבי האימון (מקסימום 10):







איך להריץ את הקוד:

1. יש לחלץ את הקובץ genderProject.ipynb מה-Zip.
2. יש להעלות את הקובץ אל Google Colab
3. יש לשנות את שם המשתמש והסיסמא למשתמש השייך לך באתר Kaggle, אם אין כזה- יש ליצור אחד.
4. יש להוסיף GPU – אחרי שלב 2 ללכת לתווית runtime ⇐ change runtime type ⇐ change hardware to gpu.
5. Run the Code

סיכום ומסקנות :

סיווג מגדר הוכח כתכונה שימושית המספקת יישומים פוטנציאליים רבים בראייה ממוחשבת וביומטריה. חקרנו שיטות מיצוי וסיווג ביחס לסיווג מגדרי , הצגנו שיטות סיווג מגדר מנקודת המבט של מיצוי וסיווג תכונות, לצד שתי שגרות עיבוד מקדמות: זיהוי פנים ונורמליזציה.
לאחר מכן השיטות הוערכו בתרחישים שונים באמצעות מאגרי מידע ציבוריים. את מאגרים אלה שנינו על מנת להתאים את הזיהוי הפנים לכל תרחיש, הגדלנו את מספר התמונות, שנינו צבעים, שנינו זוויות צילום ויצרנו מאגר חדש גדול יותר וחזק יותר אשר שיפר משמעותית את אחוז הדיוק בפרויקט, כמו כן בשיטה זו צמצמנו את אחוז הרעשים בדאטה.
כמו כן השתמשו ברשת נוירונים למציאת תכונות הבדלה בין המגדרים.
לסיכום באמצעות זיהוי פנים אפשר לפתח את פרויקט זה לפרויקטים נרחבים יותר אשר יעזרו לממשלות ולאנושות כולה לפתח דברים יעילים אשר יכולים לעזור בהרבה מאוד תחומים, אך חשוב לשים לב שלא לפגוע בזכויות האזרח במהלך שימוש מזהה הפנים.