דו"ח סיום פרוייקט Kaggle – Dog Breed Identification

מגישים: דור קליינשטרן 204881692

דרור פזו 318689049

מבוא:

לצורך הפרויקט בחרנו את האתגר של סיווג כלבים לפי הגזע שלהם. האלגוריתם מקבל כקלט תמונה של כלב, ומוציא כפלט את הסיווג של הגזע של הכלב, מתוך 120 גזעים אפשריים.

ה – Dataset שהשתמשנו בו לצורך האתגר הוא Stanford Dog Dataset – תת קבוצה של ImageNet Dataset. לכן, לא היינו צריכים לאמן רשת מהתחלה, אלא לבחור רשת שאומנה לסווג אובייקטים ב – ImageNet Dataset, ולבצע לה Fine Tuning כדי שתוכל לסווג כראוי את גזעי הכלבים.

פתרונות קודמים לבעייה:

למשימה זו Kernels רבים ב - Kaggle שנעשה בהם שימוש ברשתות מסוגים שונים - בעיקר VGG, ResNet, InceptionV3. הספריות העיקריות שנעשה בהן שימוש הן PyTorch ו - Fast.ai. היו גם מימושים ב - TensorFlow וב - Caffe.

נתייחס למספר מימושים קודמים:

1. Dog breed test with fastai - ב - Kernel זה ביצעו fine tunning לרשת resnet 101 ע"י שימוש בספרייה fast.ai. רשת ה - resnet אומנה ע"י imagenet dataset. פונקציית ה - loss שבה השתמשו היא categorical cross entropy, פונקציית loss נפוצה בשימוש ברשתות סיווג, וגם אנחנו השתמשנו בה. הפעילו 2 epochs על הרשת והגיעו ל - validation loss = 0.44 ו - accuracy = 0.8693. לאחר שאימנו את הרשת עם data נוסף שנוצר ע"י data augmentation, הגיעו ל - loss = 0.39 ו - validation accuracy = 0.889.

   קישור לעבודה:
   https://www.kaggle.com/stefanbuenten/dog-breed-test-with-fastai

2. Xception, InceptionV3 Ensemble methods - ב - Kernel זה בדקו ensembling methods שונות על הרשתות Xception ו - InceptionV3 לאחר שהביצועים על כל רשת בנפרד לא היו טובים. שיטות ה - Ensembling הן - ensembling by average ו - ensemble input features. לשתי השיטות היה loss די דומה - 0.05. המימוש היה ע"י Keras.

קישור לעבודה:

https://www.kaggle.com/robhardwick/xception-inceptionv3-ensemble-methods

3. Dog breed identification using fastai(resnet) - ב - Kernel זה ביצעו fine tunning לרשת ResNet50 ואימנו אותה על ה - Dataset של הכלבים. המימוש היה ע"י ספריית fast.ai.

קישור לעבודה:

https://www.kaggle.com/raajtilaksarma/dog-breed-identification-using-fastai-resnet

יש כמובן עוד עבודות, אך ברובן אימנו רשת בודדת - בעיקר ResNet למיניהן, InceptionV3, Xception ו - VGG. רוב המימושים נעשו ע"י הספריות keras או fast.ai.

העבודה שלנו:

פירקנו את הפרויקט לשלושה חלקים פשוטים יותר, שכל אח -

1. מכיוון שעבדנו עם PyTorch, יכלנו להעזר בספריה המובנת של ImageFolder ו DataLoader, אך שניהם צריכים שהתמונות יהיו בתת תיקיות שהשם שלהן הוא ה label שלהן, אך המידע ש kaggle סיפקו היה מהצורה של כל התמונות בתיקייה אחת, וקובץ csv שמכיל את ה label של כל קובץ. לכן הדבר הראשון שעשינו זה לממש מתודה sort שמביאה את המידע כפי שקיבלנו אותו מ kaggle ומביאה אותו לצורה שבא נוח לעבוד איתו עם PyTorch.
כמו כן נתקלנו בבעיה שלא כל התמונות בגודל אחיד ושאין מספיק תמונות מכל סוג, אך הבעיה הזאת הייתה פשוטה יחסית כי ישנו הרבה קוד שעזר לנו לבצע ארגומנטציה וכן ביצוע טרספומציות כדי להביא את כל המידע לאותן מימדים (לפי הדרישות של כל רשת) ולנרמל.
2. כעת כשיש לנו את המידע בגודל הנכון ובפורמט הנכון אפשר להתחיל לעשות fine tuning. כפי שניתן לראות בקוד, בנינו מחלקה בשם FineTuning שמקבלת לאיתחול שלה את תיקיית ה root של כל המידע, את שם המודל שרוצים לאמן (רשימה מלאה בהמשך), מספר ה classים שיהיו בסופו של דבר (במקרה שלנו יש 120), batch size, ושם התיקייה שבה יש את תיקיית ה train ותיקיית ה validation כפי שהכנו את המידע שלנו מבעוד מועד. בעת יצירת אובייקט מהמחלקה, טוענים את כל המידע לפי הפרמטרים שהתקבלו וכן מורידים וטוענים את המודל שעליו מבצעים fine tuning וכמובן מגדירים אוטומוטית את השכבה האחרונה בכל מודל שאותה אנו רוצים לאמן, וכל שאר השכבות "מוקפאות". ניתן במחלקה גם לשנות את ה optimizer הדיפולטי (שהוא SGD) לכל אחד שרוצים. הוספנו מתודה ששומרת את המודל, טוענת אותו וכמובן שיש מתודה שמאמנת את המודל כתלות במספר epochs.
בנוסף כשהקוד מאמן את המודל, אנו בודקים אם לאחר ה epoch הנוכחי ה validation accuracy יורד, אם כן אז הוא מתעלם מהאימון הנוכחי, ומחזיר את המודל אחורה לכפי שהיה מקודם (עם validation accuracy גדול יותר).
3. לבסוף לאחר שיש לנו מודל מאומן נותר לבדוק עבור המידע של ה test של kaggle לצורה שהם רוצים, כלומר קובץ csv אחד ארוך. בשיבל זה מימשנו מתודה kaggle_csv שמקבל את תיקיית ה test ושם קובץ ה csv המבוקש ומדרג את תוצאות המודל בדיוק כפי שהם רוצים, ובעזרתו אפשר למדוד את הביצועים של המודלים השונים

דיון:

בעבודתנו ביצענו fine tunning לרשתות שונות, עם optimizers ו - learning rate שונים. חשוב לציין שלקחנו את התוצאות הטובות ביותר עבור כל optimizer, לכן ה - learning rates לא שווים. learning rates קטנים מדי הביאו ללימוד איטי מדי, ו - learning rates גבוהים מדי גרמו להתבדרות ול - loss הולך וגדל.

להלן מספר טבלאות שמסכמות את התוצאות עבור רשתות שונות (אימנו כל רשת כ - 15 epochs):

Inception V3:

Validation Accuracy	Loss	Optimizer	Learning Rate
0.874	**0.43**	**Adam**	**1e-3**
0.8729	0.4329	Adadelta	5e-1
0.844	0.7	RMSProp	5e-3

VGG16:

Validation Accuracy	Loss	Optimizer	Learning Rate
0.8152	0.5458	Adam	1e-3
0.8142	0.544	Adadelta	5e-1
0.8017	0.658	RMSProp	5e-3

ResNet50:

Validation Accuracy	Loss	Optimizer	Learning Rate
0.84	0.53	Adam	1e-3
0.8326	0.58	Adadelta	5e-1
0.8259	0.62	RMSProp	1e-3

בנוסף ראינו כי לאורך הרבה מאוד מודלים שונים, לא הצלחנו כל כך לרדת מ loss של ~0.43 לא חשוב מה עשינו.

כמו כן החיסרון הכי בולט במהלך הפרויקט הוא שיש לנו רק ~80 תמונות מכל גזע, דבר שמאוד הקשה עלינו כי זה נחשב מעט מאוד מידע, אך התגברנו על הקושי הזה על ידי שימוש חכם ב fine tuning עבור רשתות שכבר אומנו בעבר על כלבים.

דברים שלא הספקנו לעשות אבל חשבנו עליהם - לחקור יותר לעומק על רשתות שכבר אומרנו ולבדוק את ה data של כל אחת ולראות איפה יש דגש יותר חזק על כלבים. חשבנו גם לעשות data augmentation עם GAN אך מחוסר זמן וידע מעשי בתחום בסוף החלטנו שלא לעשות זאת.

סיכום:

לסיכום קיבלנו תוצאה מרשימה של validation accuracy 87% ו - loss 0.43 עבור inception V3 לאחר שבדקנו סוגים שונים של רשתות עם פרמטרים שונים ו optimizerים שונים וזה הכי טוב שהצלחנו.

למדנו הרבה מאוד מהפרויקט הזה, ואנו הכי שמחים שכעת יש לנו קוד מאוד מודלרי שאנו יכולים ליישם אותו להרבה תחומים שונים בנושא של ראיה ממוחשבת על ידי שינוי קל מאוד בקוד.

רעיונות לפרויקט המשך: להשתמש במודל שלנו על מנת לעשות אפליקציה שמצלמת בזמן אמת תמונה של כלב ומודיעה למשתמש מה הגזע שלו.

קישור למודל המאומן + תוצאות הריצה.