למידת מכונה – פרויקט גמר

מגישים:

זאב פישר 318960242
יוליה כץ 324385509

נושא הפרויקט: סיווג תמונות של חיות / כלי תחבורה

מאגר הנתונים:

מאגר הנתונים עליו אנו עובדים הוא CIFAR-10.

הוא מורכב מ 60,000 תמונות המחולק ל 50,000 train 10,000 test.

התמונות מחולקות לנושאים הבאים: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck כאשר לכול אחד יש 5000 תמונות.

כול התמונות במאגר הם באותו הגודל 32x32

במימוש שלנו צמצמנו את כמות התמונות כך שלכול סיווג יש:

Train: 1000 images

Test: 100 images

סה"כ יש לנו 10,000 train 1,000 test

רעיון הפרויקט:

בפרויקט זה לקחנו 4 כלים שונים, כול ארבעת הכלים יעבדו על אותו מאגר מידע ובדיוק אותם תמונות לכול השלבים של אימון ובדיקת יעילות הכלים.

מטרת המחקר:

בהינתן מאגר התמונות נרצה לאמן כול כלי עם אותו מאגר ונרצה לבדוק את יעילות הסיווג שלו על תמונות שהכלי עוד לא ראה.

בהמשך לכול אחד מארבעת הכלים נפרט כיצד הוא מסתכל על התמונה וכיצד הוא לומד את הפיצ'רים של התמונה.

הנחות מיוחדות:

כמו שאמרנו כבר כול ארבעת הכלים עובדים בדיוק עם אותם התמונות לאמן ולבדוק את יעילות הסיווג.

כלי המחקר:

AdaBoost (Adaptive Boosting) using Decision Tree

מבוא:

Adaboost הוא אלגוריתם למידה אנסמבלי (המשלב מספר מודלים אינדבידואליים) המשמש למשימות סיווג ורגרסיה. מטרתו העיקרית היא לשלב מספר מסווגים חלשים, בדרך כלל מודלים פשוטים, למסווג או מנבא חזק אחד. המטרה היא לשפר את הביצועים החזויים על פני מסווג חלש יחיד על ידי התמקדות איטרטיבית במקרים שסווגו בצורה שגויה בנתוני האימון.

Adaboost עובד באמצעות תהליך איטרטיבי. בתחילה, לכל מופע אימון מוקצה משקל שווה. מסווג חלש ,לרוב עץ החלטות עם עומק מוגבל (גדם), מאומן על הנתונים המשוקללים הללו. לאחר האימון , Adaboostמעריך את הביצועים של המסווג החלש. לאחר מכן הוא מקצה משקלים גבוהים יותר למקרים שסווגו בצורה שגויה, מה שמניע את המסווג החלש הבא להתמקד יותר במקרים הקשים הללו. תהליך זה חוזר על עצמו עבור מספר קבוע מראש של איטרציות או עד להשגת מודל מושלם. המודל הסופי משלב את התחזיות של כל הלומדים החלשים, כאשר תרומתו של כל לומד חלש משוקללת לפי ביצועיו

Adaboost עם עצי החלטה כמסווגים חלשים יעיל במיוחד עבור משימות סיווג בינארי. הוא עובד היטב עם מאגרי נתונים המכילים שילוב של תכונות מספריות וקטגוריות. עם זאת, ניתן ליישם את Adaboost גם על משימות רגרסיה והוא יכול לעבוד עם סוגים שונים של נתונים. יכולת ההסתגלות והביצועים החזקים שלו הופכים אותו למתאים למגוון רחב של יישומי למידת מכונה, במיוחד כאשר מתמודדים עם מערכות יחסים מורכבות בתוך הנתונים.

במימוש שלנו:

כדי להשתמש ב Adaboost אנחנו קודם כול צריכים לתת לו חוקים חלשים שהוא יוכל ליצור חוק חזק. למטרה זו ננסה להוציא מתוך התמונה פיצ'רים שנוכל לתת ל Adaboost ללמוד.

אנחנו עושים זאת על ידי המרת התמונה ל NumPy array ולאחר מכן ממירים את הצורה הזו לווקטור על ידי flatten ואת הווקטור הזה אנחנו נותנים לכלי ללמוד.

שימוש בשיטה זו בלבד נתנה לנו תוצאות נמוכות. לכן הוספנו פונקציה חדשה שתקרא את התמונות רק שהפעם אנחנו משתמשים במודל מוכן שנקרא VGG16 אנחנו לוקחים ומעבירים אותו במודל המוכן על מנת להוציא פיצ'רים

לאחר מכאן אנחנו ממירים את הסיווגים ל-integers על מנת להתאים אותם ללמידת מכונה

ולבסוף אנחנו בודקים את יעילות הסיווג.

שימוש ב VGG16 נותן לנו: Accuracy: 0.59

שימוש בעיבוד בסיסי של התמונה נותן לנו: Accuracy: 0.43

SVM - Support Vector Machine

מבוא:

SVM הוא אלגוריתם למידה מפוקח המשמש למשימות סיווג ורגרסיה. המטרה העיקרית של SVM היא למצוא את המישור האופטימלי המפריד בצורה הטובה ביותר בין המחלקות השונות במרחב התכונות. בסיווג SVM ,שואפת ליצור גבול החלטה שממקסם את המרווח בין מחלקות, ולמקסם את ההפרדה בין נקודות נתונים של מחלקות שונות. ניתן להשתמש ב SVM-גם למשימות רגרסיה, שבהן הוא שואף למצוא מישור המתאים ביותר לנתונים תוך מזעור שגיאות.

SVM הופך את נתוני הקלט למרחב בעל ממדים גבוהים יותר שבו ניתן למצוא היפר מישור כדי להפריד בין המחלקות. הוא עושה זאת באמצעות פונקציית ליבה, המאפשרת ל SVM-למפות באופן מרומז את תכונות הקלט לתוך מרחב ממדי גבוה יותר מבלי לחשב את הטרנספורמציה באופן מפורש. ברגע שהוא נמצא במרחב בעל הממדים הגבוהים יותר SVM ,מוצא את המישור הממקסם את המרווח בין המחלקות. נקודות הנתונים הקרובות ביותר להיפר מישור נקראות וקטורי תמיכה, והן קובעות את המיקום והכיוון של המישור SVM .מבקש למצוא את המישור הממזער את שגיאת הסיווג תוך מקסום השוליים, מה שהופך אותו לעמיד בפני חריגים ויעיל בטיפול במערכי נתונים מורכבים.

SVM יעיל גם עבור מאגרי נתונים הניתנים להפרדה ליניארית וגם עבור מאגרי נתונים שלא ניתנים להפרדה ליניארית. הוא עובד היטב עם נתונים במימדים גבוהים והוא שימושי במיוחד כאשר מספר התכונות גדול ממספר הדגימות SVM .מתאים למשימות סיווג עם תוצאות בינאריות ורב מחלקתיות. הוא יכול להתמודד עם נתונים מספריים וקטגוריים, מה שהופך אותו למגוון עבור סוגים שונים של מאגרי נתונים. עם זאת, ייתכן ש SVM-לא יתאים למאגרי נתונים גדולים מאוד בשל המורכבות החישובית שלו, והוא עשוי לדרוש בחירה קפדנית של היפרפרמטרים לביצועים מיטביים. בסך הכל , SVMמתאים למשימות בהן רצויה הפרדה ברורה בין מחלקות וכאשר עוסקים במאגרי נתונים בגודל בינוני.

במימוש שלנו:

הבעיה העיקרית שנתקלנו בה כאשר השתמשנו בכלי זה הוא זמן הריצה הגדול מאוד ביחס לשאר הכלים (מעל 9 שעות).

- לאחר קריאת התמונות לקחנו את התמונה שהייתה מיוצגת כמטריצה והשתמשנו ב-Flattening להפוך אותה לווקטור יחיד.
- לאחר מכאן עשינו Scaling בכך שהפכנו את הממוצע שלהם ל0 ואת סטיית התקן שלהם ל 1 תהליך זה משפר את הביצועים בשביל ה SVM בכך שהוא מבטיח שכול התכונות יהיו באותו הטווח ומקל על האלגוריתם לזהות דפוסים ולסווג את הנתונים בצורה מדויקת
- ולבסוף השתמשנו ב PCA (Principal Component Analysis) לצמצם את ממדי הנתונים ולצמצם רעש

אימון המודל:

על מנת לאמן את המודל השתמשנו ב Grid Search הפרמטרים שהגדרנו לצורך הבדיקה הם

C: פרמטר המאזן בין התאמה טובה לנתונים עליהם המודל מאומן לבין מניעת .overfitting ערכים אפשריים: 0.1, 1, 10.

Kernel: פונקציית גרעין שמשנה את המרחב בו SVM עובד. ערכים אפשריים:

- Poly הפרדה פולינומית
- Linear הפרדה לינארית
- Rbf(Radial Basis Function) מתאימה למקרים בהם הנתונים לא לינאריים כלל.

Gamma: פרמטר המגדיר את האופן שבו כל דוגמה בודדת תשפיע על ההחלטה של המודל. ערכים אפשריים: scale and auto אלה פשוט צורות חישוב שונות כול אחת עם הפונקציה שלה. הם משמשים בגרעין של SVM והם משפיעים על האופן שבו כול דוגמה בודדת משפיעה על הקו המפריד.

לבסוף אנחנו משתמשים ב 2 כלים נוספים:

- Cross-Validation: כדי להעריך את הביצועים של כל צירוף היפר-פרמטרים Grid Search , משתמש ב cross-validation-עם 3 קיפולים .(cv=3) זה אומר שהנתונים יחולקו לשלושה קיפולים, כאשר בכל פעם קיפול אחד ישמש לבדיקת המודל והשאר לאימון. כך, המודל נבדק שלוש פעמים על קיפולים שונים של הנתונים, מה שמסייע להבטיח שהביצועים המוערכים של המודל הם מדויקים ולא תוצאה של חלוקה מקרית של הנתונים.
- Best Estimator: לאחר שהושלמה בדיקת כל הצירופים האפשריים Grid Search ,בוחר את המודל שהראה את הביצועים הטובים ביותר על סט הוולידציה.(validation set) . המודל הזה, שנקרא best_estimator_,הוא זה שמוחזר על ידי הפונקציה, כך שניתן יהיה להשתמש בו להמשך תהליך התחזיות (prediction) על נתונים חדשים.

בסופו של דבר אנחנו מקבלים Accuracy: 45.20%

כול טבלת התוצאות מוצגת בסוף.

Random Forest:

מבוא:

Random Forest הוא אלגוריתם למידה אנסמבלי (המשלב מספר מודלים אינדבידואליים) רב תכליתי המשמש לסיווג, רגרסיה ומשימות אחרות. מטרתו היא לשפר את הביצועים של עץ החלטה בודד על ידי יצירת "יער" של עצי החלטה ושילוב תחזיותיהם Random Forest. שואפת להפחית overfitting ולהגביר את החוסן על ידי אימון עצי החלטה מרובים על תת קבוצות אקראיות של נתוני האימון ושילוב תחזיותיהם באמצעות הצבעה או מיצוע (ממוצע התוצאות).

Random Forest עובד על ידי יצירת אנסמבל של עצי החלטה. במהלך האימון, עצי החלטה מרובים גדלים תוך שימוש בקבוצות משנה שונות של נתוני האימון והתכונות. כל עץ מאומן באופן עצמאי, ובכל פיצול נלקחת בחשבון תת-קבוצה אקראית של תכונות, מה שעוזר לקשור את העצים ולהפחית overfitting. במהלך חיזוי, תחזיות העצים הבודדות משולבות באמצעות מיצוע (לרגרסיה) או הצבעה (לסיווג) כדי לייצר את התחזית הסופית. גישת אנסמבל זו מביאה בדרך כלל למודל מדויק וחזק יותר בהשוואה לעץ החלטות בודד.

Random Forest מתאים היטב למגוון רחב של מערכי נתונים ומשימות. זה יכול לטפל גם בבעיות סיווג וגם בבעיות רגרסיה, והוא יעיל עם נתונים במימד גבוה המכילים מאפיינים מספריים וקטגוריים כאחד Random Forest. עמיד בפני חריגים, ערכים חסרים ונתונים רועשים, מה שהופך אותו למתאים עבור מערכי נתונים בעולם האמיתי עם דרגות שונות של איכות. הוא יכול להתמודד עם מערכי נתונים לא מאוזנים והוא נוטה פחות ל overfitting-בהשוואה לעצי החלטה בודדים. עם זאת, ייתכן ש forest-Random אינו הבחירה הטובה ביותר עבור משימות הדורשות מודלים ניתנים לפירוש, מכיוון שמכלול עצי ההחלטה יכול להיות מורכב לפירוש.

במימוש שלנו:

דומה ל Adaboost גם פה השתמשנו ב2 גישות על מנת לנסות להגיע לסיווג טוב של הנתונים.

שיטה ראשונה והבסיסית היא לקרואת את התמונות המיוצגות על ידי מטריצה ולהפוך אותם לווקטור על ידי שימוש ב flatten כאשר את הווקטור הזה אנחנו יכולים להכניס לתוך Random Forest והוא ילמד אותו.

בהשוואה לכך שיטה נוספת היא להשתמש במודל ה 16VGG שבמקום להחזיר לנו ייצוג של התמונה הוא מחזיר לנו ייצוג מטריצה של הפיצ'רים החשובים בתמונה את אותם פיצ'רים אנחנו ממרים לווקטור יחיד שוב על ידי flatten ואת הווקטור הזה אנחנו מכניסים לRandom Forest

ההבדל בשיטות אלו הוא כמובן לא בצורה ב Random Forest עובד על ווקטור אלה כמובן באיכות הווקטור עליו אנחנו עובדים.

כאשר אנחנו משתמשים ב VGG16 אנחנו מקבלים Accuracy: 0.59.

ביחד ל Accuracy: 0.43 כאשר אנחנו משתמשים בצורה המפושטת של התמונה.

CNN (Convolutional Neural Networks)

רשתות נוירונים קונבולוציוניות (Convolutional Neural Networks או בקיצור CNN) הן סוג של רשתות נוירונים עמוקות שמתמקדות בעיבוד נתונים בעלי מבנה רשת ,(grid-like topology) כגון תמונות. בניגוד לרשתות נוירונים רגילות CNNs ,(Fully Connected Neural Networks) מתמקדות בזיהוי תבניות מרחביות בתמונות על ידי שימוש במסננים (filters) ובשכבות קונבולוציה שמחלצות מאפיינים (features) ממבני התמונה באופן אוטומטי.

עקרונות פעולה של:CNN

1. **שכבות קונבולוציה :(Convolutional Layers)** השכבות הקונבולוציוניות הן המרכיב המרכזי ב .CNNs-בשכבות אלו, מסננים (kernels) נעים על פני התמונה וקובעים את עוצמת החיבור בין הפיקסלים השכנים. התוצאה היא מפות תכונות (feature maps) שמייצגות תבניות שונות בתמונה כמו קצוות, פינות, וטקסטורות.
2. **שכבות מיצוי :(Pooling Layers)** שכבות מיצוי משמשות להקטנת הממדים של מפות התכונות שנוצרו בשכבות הקונבולוציה, ובכך להפחית את כמות המידע שהמודל צריך להתמודד איתו, מבלי לאבד מידע חשוב. סוג נפוץ של מיצוי הוא ,Max Pooling שבו נבחר הערך המקסימלי בכל אזור במפת התכונה.
3. **שכבות צפופות :(Fully Connected Layers)** לאחר מעבר דרך מספר שכבות קונבולוציה ומיצוי, המידע מומר לווקטור חד-ממדי באמצעות שכבת ,Flatten ולאחר מכן מועבר לשכבות צפופות, אשר מבצעות את תהליך הסיווג הסופי.
4. **שכבת הפלט :(Output Layer)** בשכבת הפלט, נעשה שימוש בפונקציית softmax כדי להפיק הסתברות לכל אחת מהקטגוריות האפשריות, כאשר הקטגוריה עם ההסתברות הגבוהה ביותר נבחרת כתוצאה הסופית.

במימוש שלנו:

לאחר קריאה של כול התמונות במקרה הזה בניגוד לשאר המודלים אחד השלבים החשובים הוא לעשות shuffle לתמונות שלב זה חשבו בכללי כאשר אנו מאמנים מודל אך בשאר הכלים שמתנו לב כי הוא פחות משמעותי חוץ מCNN בו הוא חשוב ביותר.

לאחר מכאן אנחנו מכינים את הדאטה וממירים כול תמונה לערך בין 1-0

וממרים כול תווית לפורמט קטלוג כלומר כול תווית מיוצגת כווקטור באורך השווה למספר הקטגוריות עם 1 במקום המתאים לקטגוריה

שלבים אלו מכינים את הדאטה לשלב אימון המודל. אך קודם כול אנחנו צריכים לבנות את המודל.

המודל שלנו בנוי מרצף של שכבות

:**layers.Conv2D** שכבות קונבולוציה שמפיקות מפות תכונות (feature maps) מהתמונות. בכל שכבה יש עלייה במספר הפילטרים (32, 64, 128) להגדלת העומק והמורכבות של הייצוגים הנלמדים.

:**layers.MaxPooling2D** שכבות מיצוי (pooling) שמפחיתות את הממדיות של מפות התכונות, תוך שמירה על המידע החשוב.

:**layers.Flatten** מרה של מפות התכונות לווקטור חד-ממדי, כדי שניתן יהיה להכניסו לשכבות צפופות.

layers.Dense: שכבות צפופות שמבצעות סיווג על סמך התכונות שהופקו. השכבה האחרונה משתמשת בפונקציית softmax להפקת הסתברות לכל קטגוריה.

כדי להימנע מ overfitting מקרה בו המודל לומד יותר מדיי פרטים ונהיה מדויק מדיי בכך שהוא לא נותן מספיק מקום לimperfections אנחנו משתמשים ב:

ReduceLROnPlateau קריאת חזרה המפחיתה את קצב הלמידה כאשר ביצועי המודל אינם משתפרים במשך מספר אפוקים .(patience) זה עוזר להימנע מאימון יתר.(overfitting)

ולבסוף בשיטה זו אנחנו מגיעים לתוצאה הטובה ביותר שלנו של

Test accuracy: 0.6729999780654907

הערה חשובה כאשר המודל למד שמנו לב שבהתחלה בעקבות כמות השכבות הגדולה שהייתה לנו נתקלנו במצב של overfitting ודווקא צורה מופשטת יותר עם פחות שכבות ב CNN נתנה לנו תוצאה טובה יותר עם זאת עדין יש קצת overfitting אנחנו מניחים שזה בגלל גודל התמונה המצומצם שמגביל את יכולת הלימוד של המודל.

סיכום התוצאות ומסקנות:

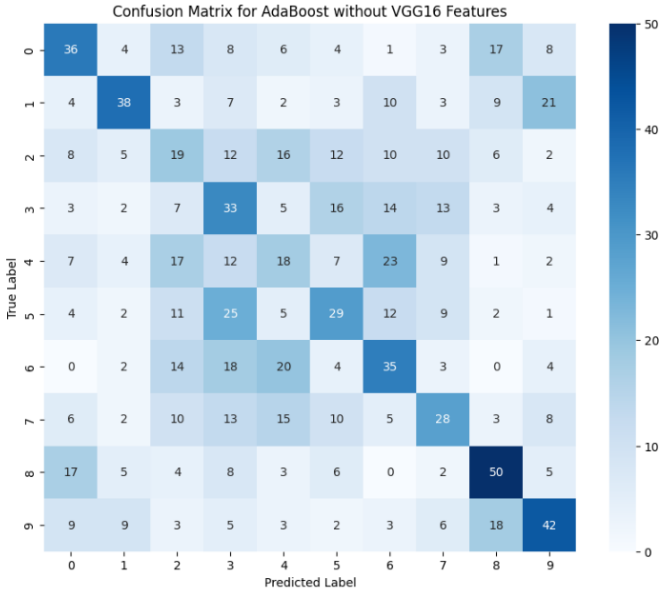
ראשית ניתן למצוא קישור לכול קטעי הקוד בגיט הבאה GitHub

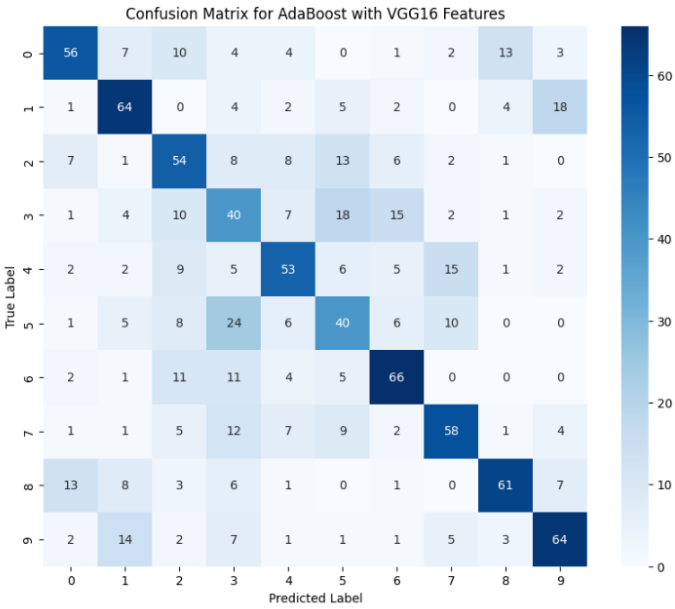
כמו שכבר הזכרנו לפרויקט זה השתמשנו בספרית CIFAR-10 ספריה גדולה מאוד המכילה 60 אלף תמונות סה"כ בעלי 10 תיוגים\סיווגים שונים. בעיה אחת בספריה הזו היא עצם הגודל שלה לוקח הרבה מאוד זמן לעבד ולעבוד עם ספריה כול כך גדולה ולכן היינו חייבים לצמצם אותה. בעיה שניה קשורה לגודל התמונות כול תמונה מיוצגת בגודל שווה של 32X32 להקל על גודל ספריית התמונות אך זה בא על חשבון כמות המידע בתמונה, ככול שהתמונה גדולה יותר ואיכותית יותר היא מכילה יותר מידע בתוכה ומאפשרת ניתוח טוב יותר והוצאת פיצ'רים טובים יותר מיתוך התמונה. לכן גם במודל הכי טוב שלנו שהוא CNN הצלחנו להגיע ל67% דיוק בלבד.

תוצאות:

AdaBoost: VGG16 Accuracy: 0.5510 => 55%

Basic Accuracy: 0.3280 => 32%





האלכסון הראשי מסמל את התמונות שתויגו נכון אם ניסכום כול שורה סה"כ אנחנו צריכים להגיע ל 100 תמונות.

ניתן לראות כי בתמונה התחתונה שמייצגת את המודל VGG16 המספרים באלכסון הראשי גדולים יותר כלומר המודל הצליח לסווג יותר תמונות בצורה נכונה וכמות התמונות שהוא הצליח אף תואם בממוצע ל Accuracy שקיבלנו

לאומת זאת כאשר נסתכל על התמונה העליונה נראה פיזור הרבה יותר גדול של מספרים ואף מספרים נמוכים משמעותית באלכסון הראשי מה שמראה על סיווג הרבה יותר נמוך

airplane 0

automobile 1

bird 2

cat 3

deer 4

dog 5

frog 6

horse 7

ship 8

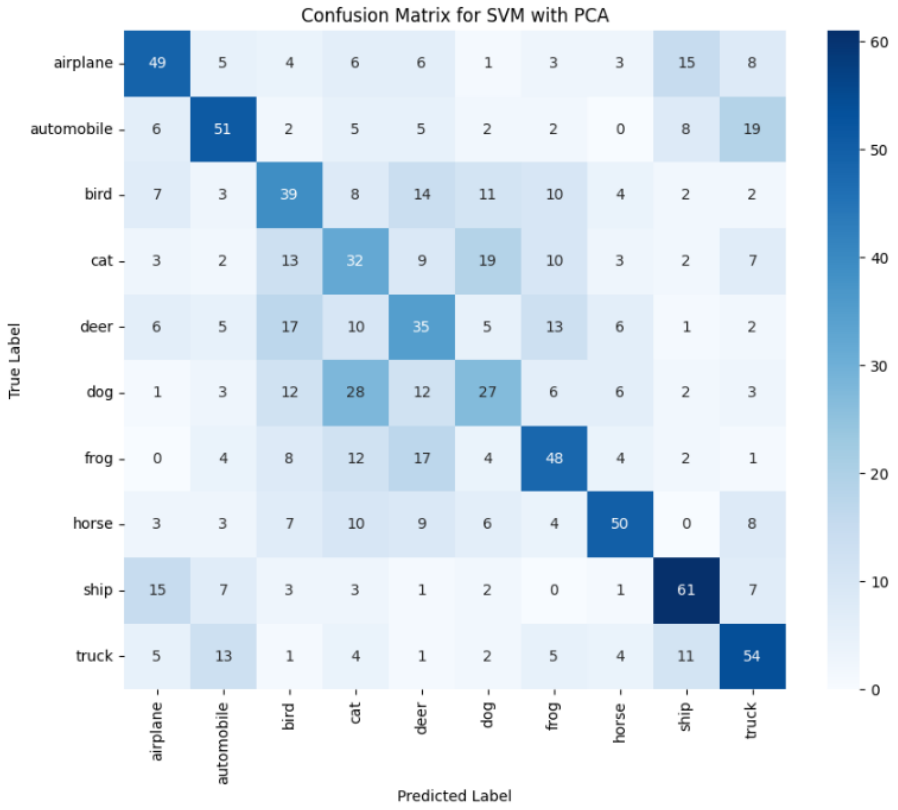
truck 9

SVM:

```
Accuracy: 44.60%
Classification Report:
              precision    recall  f1-score   support

    airplane       0.52      0.49      0.50       100
  automobile       0.53      0.51      0.52       100
        bird       0.37      0.39      0.38       100
         cat       0.27      0.32      0.29       100
        deer       0.32      0.35      0.33       100
         dog       0.34      0.27      0.30       100
        frog       0.48      0.48      0.48       100
       horse       0.62      0.50      0.55       100
        ship       0.59      0.61      0.60       100
       truck       0.49      0.54      0.51       100

    accuracy                           0.45      1000
   macro avg       0.45      0.45      0.45      1000
weighted avg       0.45      0.45      0.45      1000
```

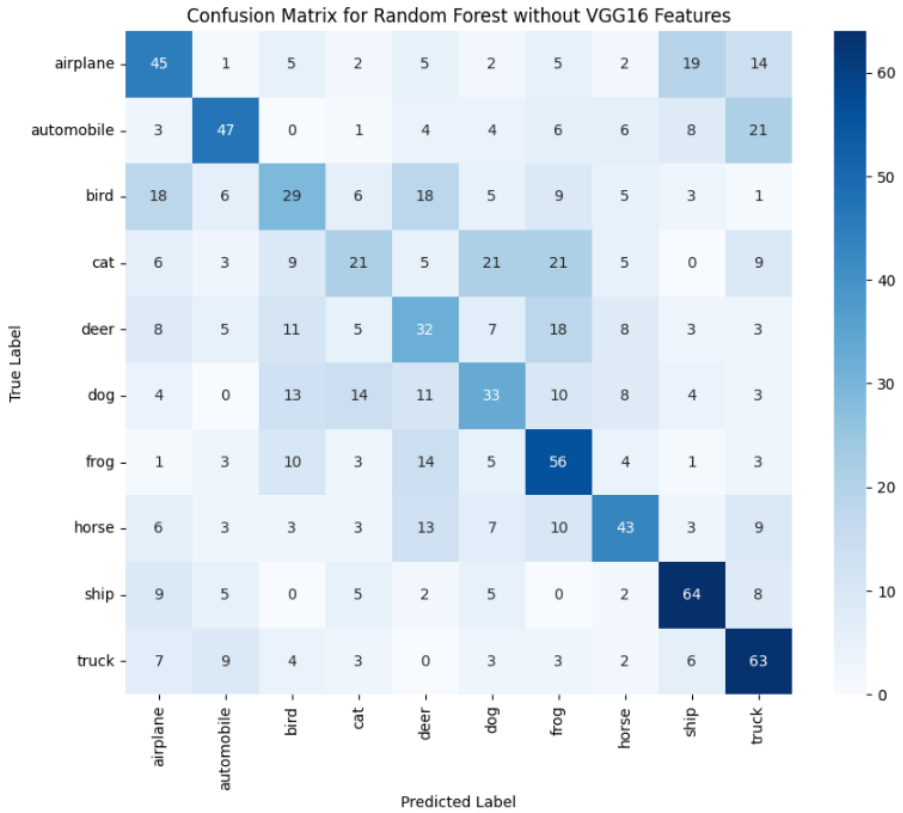


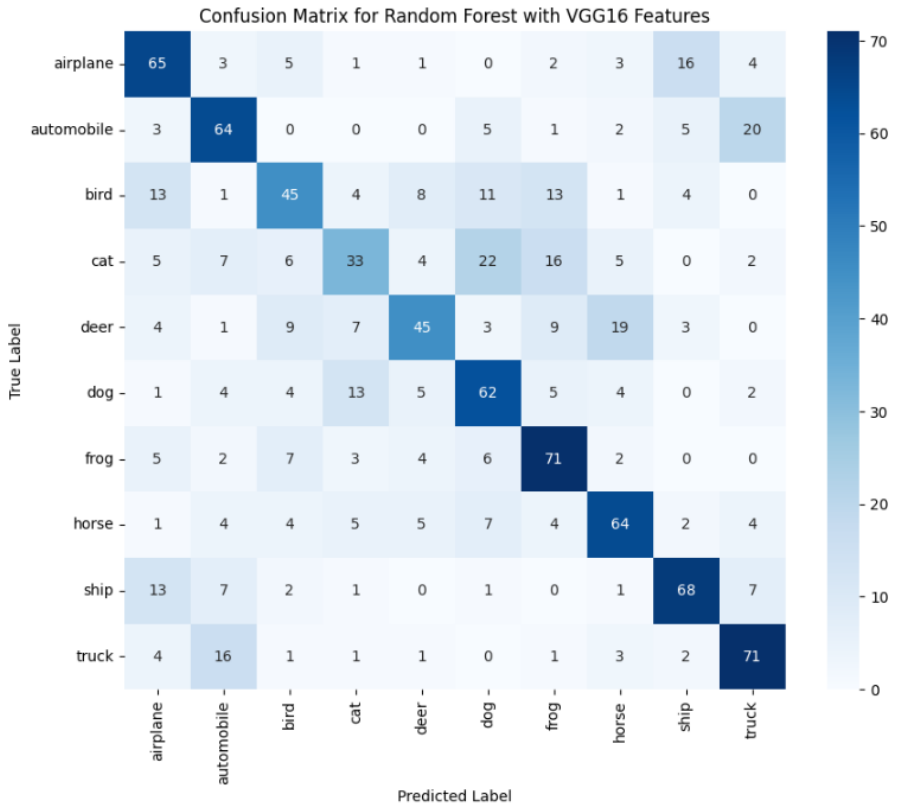
כמובן גם בתמונה הזו האלכסון הראשי של המטריצה מסמל את כמות התמונות שאכן הצלחנו לסווג כמו שצריך.

במקרה של SVM ה confusion matrix יכול להראות לנו שיש מספר תיוגים שגרמו לאחוז הדיוק לרדת כמו כלב ציפור וחתול לאומת תיוגים שהצליחו משמעותית יותר טוב כמו ספינה סוס ומשאית

RandomForest: VGG16 Accuracy: 0.59 => 59%

Basic Accuracy: 0.43 => 43%



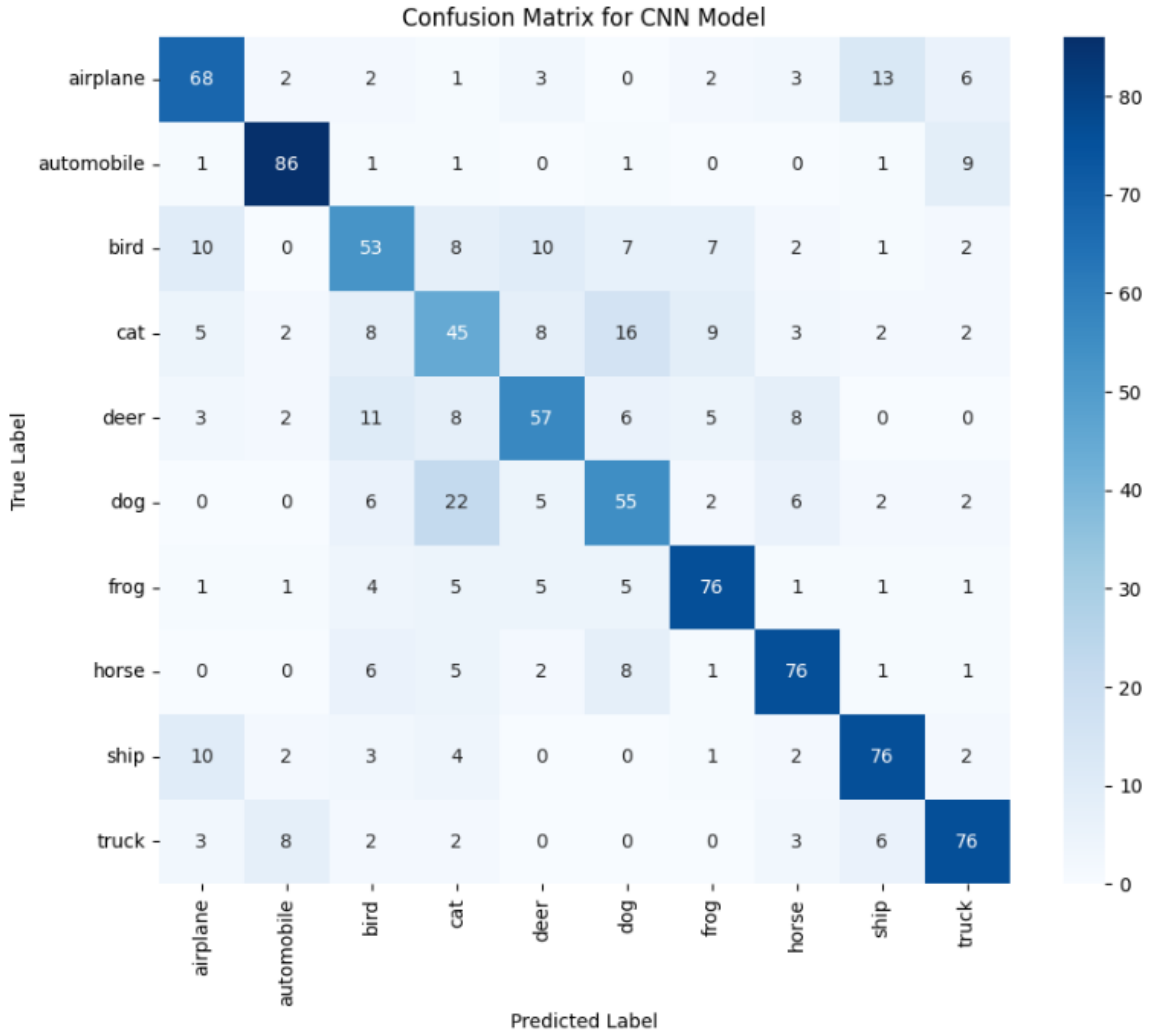


נסתכל על התוצאות של RandomForest ונראה שהם תואמות בצורה מסוימת למקרה של adaboost האלכסון ב 2 התמונות מייצג את הסיווגים המוצלחים ניתן לראות שכאשר השתמשנו ב VGG המודל שלנו הצליח משמעותית הרבה יותר טוב

ומה שניתן כבר לשים לב שיש פה דפוס של תיוגים שכול 3 המודלים שהסתכלנו עליהם עד עכשיו מצליחים בהם בצורה הרבה יותר טובה ביחד לתיוגים אחרים לדוגמה:

כול 3 המודלים עד כו הצליחו יחסית טוב בתיוג ספינה אך מתקשים יותר בתיוג ציפור בחלק מהמקרים

CNN: Accuracy: 0.6729999780654907 => 67%



כמובן שמודל ה CNN הביא לנו את התוצאות הכי טובות ולכן את המספרים הכי גדולים האלכסון הראשי ואפילו במודל הזה ניתן בכול שאת לראות שיש מספר תיוגים שבעקביות מצליחים פחות כמו בשאר המודלים.

סיכום

ניתן לראות פה כי CNN נותן את התוצאות הכי טובות בעיקר כי הוא מיועד לניתוח מידע מהסוף שיש לנו. אנחנו רואים שבמקום השני מגיע SVM שיכול להגיע לתוצאות טובות אך הוא לוקח הרבה יותר זמן בהשוואה לCNN שלקח פחות מדקה לאומת 9 שעות הסיבה היא שלוקח ל SVM הרבה יותר זמן לנתח את המידע ולהוציא את הפיצ'רים הדרושים (בקישור לקוד ניתן לראות בדיוק איזה חלק ב SVM לקח כולכך הרבה זמן).

במקום שלישי ניתן לראות ש RandomForest נותן תוצאות מאוד קרובות לSVM וכאשר משתמשים במודל המוכן של VGG16 הוא אפילו עוקף את SVM ומגיע לתוצאות של כמעט 60%.

במקום האחרון והמסווג הכי פחות יעיל שלנו הוא Adaboost שמגיע לסיווג של 32% כאשר אנחנו משתמשים בניתוח התמונה עצמה. אך כאשר אנחנו משתמשים ב VGG16 אנחנו כן מצליחים להגיע לתוצאות יותר טובות של 55%

אם נסתכל על כול התמונות של ה confusion matrix נוכל לראות כי כול ארבעת המודלים התקשו בצורה כזו או אחרת בסיווגים של ציפור, חתול, כלב, וצבי והצליחו דווקא בתיוגים כמו משאית ספינה וסוס. הניחוש שלנו להצלחה הזו הוא גודל האובייקט המדובר והיחס שלו בתמונה של ציפור לדוגמה יש רקע של טבע מה שמוסיף יותר פרטים ויכול לבלבל את המודל. לאומת משאית שככול הנראה תתפוס חלק יותר גדול בתמונה, כמובן גם בהתחשב בגודל התמונה הקטן כבר הסיכוי ל "רעש" נוסף בתוך התמונה נמוך יותר ולכן נקבל סיווג טוב יותר.

חשוב להדגיש ש VGG16 הוא מודל מבוסס CNN אנחנו כמובן לא משתמשים במודל הזה לסווג את התמונה אלה רק להוציא פיצ'רים ולנתח את התמונה ועל ידי נתונים אלו לאמן את המסווגים שלנו. ולכן שימוש במודל המוכן הזה משפר משמעותית את תוצאות המסווג שלנו.