مستندات پروژه دوم هوش محاسباتی

اعضای گروه: الناز محمدی، زهرا رستمی

فاز اول: Image Filtering

در این فاز ابتدا هدف طراحی تابعی است که عملیات کانولوشن را اجرا کند. در کد ما تابع apply_convolution(image, kernel) **این عملیات را انجام می‌دهد. در این تابع ابعاد کرنل محاسبه و همچنین ابعاد تصویر استخراج می‌شود.**

به طور کلی فاز اول برای موارد زیر انجام می‌شود:

1- **اعمال فیلتر** sobel **(شامل دو کرنل است که هردو به طور جدا روی تصاویر اعمال می‌شود.)**
2- **اعمال فیلتر** hog
3- **اعمال فیلتر دلخواه**
4- **اعمال فیلتر** sobel + hog
5- **اعمال فیلتر دلخواه** + hog
6- **داده خام**

پس از اعمال فیلترهای بالا روی تصاویر، ویژگی‌هایی از تصاویر استخراج شده و برای هریک بردار ویژگی ساخته شده است.

در ادامه به بررسی جزئی‌تر هر یک از فیلترها می‌پردازیم:

1- فیلتر sobel

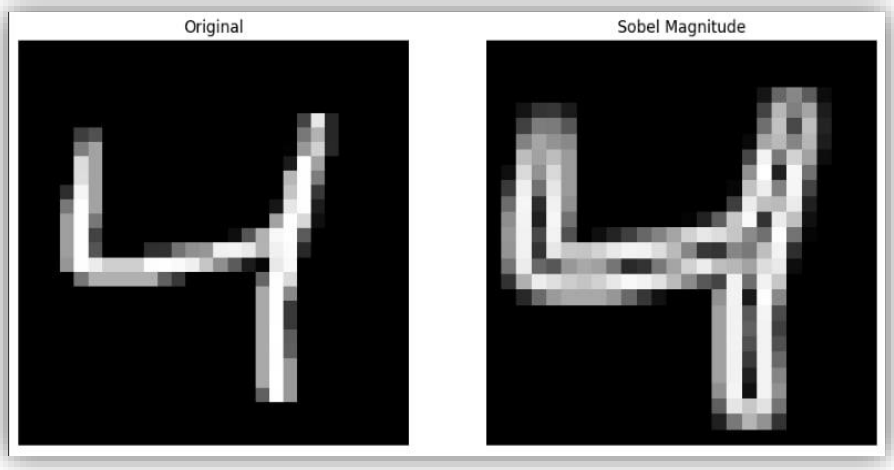
فیلتر سوبل شامل دو کرنل x , y هست که sobel x برای شناسایی لبه‌های افقی و sobel y برای شناسایی لبه‌های عمودی اعمال می‌شود. از آن‌جایی که این فیلتر لبه‌ها را شناسایی می‌کند انتظار می‌رود در خروجی ما، مرزها آشکار و قابل تشخیص باشند. (طبق عکس 1)

با ترکیب گرادیان های افقی و عمودی طبق فرمول زیر شدت کلی گرادیان برای هرپیکسل محاسبه می‌شود که این مقدار نمایانگر شدت لبه در هر نقطه از تصویر است.

$$\sqrt{{G_x}^2 + {G_y}^2}$$

با کمک gradient_magnitude.flatten() **ما بردار ویژگی‌ها را خواهیم داشت.** (feature_vector).

این فیلتر برای طبقه‌بندی دیتاست ما خوب است اما برای اعدادی که تفاوت جزئی دارند (مانند 3 و 8) ممکن است محدودیت داشته باشند و طبقه‌بندی دقیقی نداشته باشیم. همچنین با اعمال تکنیک‌هایی مثل ترکیب Sobel با thresholding (برای کاهش نویز)، یا افزایش وضوح با preprocessing می‌توان آن را بهبود داد.

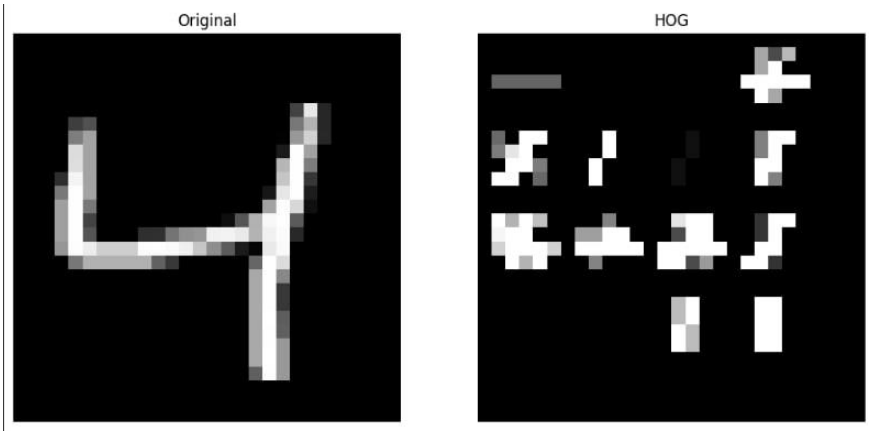


عکس 1 : اعمال فیلتر sobel بر روی تصویر و تاثیر آن

2- فیلتر hog

این فیلتر با در نظر گرفتن لبه‌ها و زوایا، الگوهای بافتی و هندسی را به دست می‌آورد. از آن‌جایی که این فیلتر جزئیات بیشتری از ساختار داخلی اعداد به دست می‌آورد برای شناسایی دقیق‌تر در اعداد مناسب است. همچنین از آن‌جایی که بازنمایی کامل‌تری از شکل دارد ، در الگوریتم‌های دسته‌بندی مثل SVM عملکرد خوبی دارد. برای بهبود این فیلتر نیز می‌توان از مقیاس‌های متفاوت و اندازه‌های بهینه برای cell و block استفاده کرد.

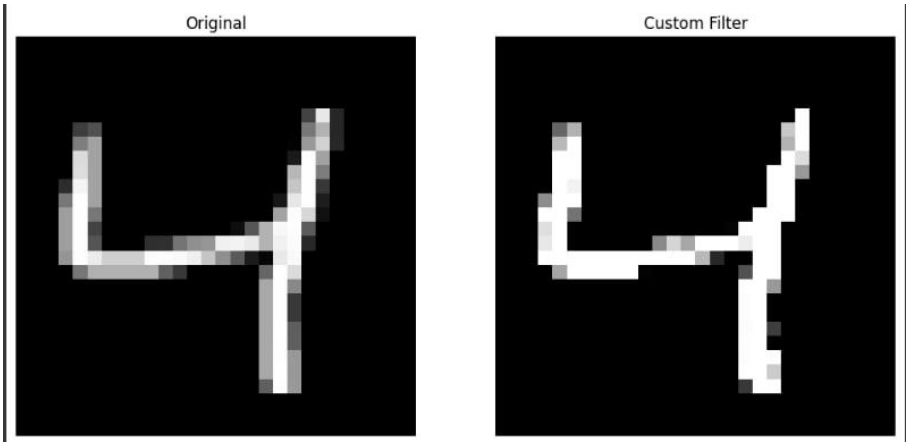
درتابع extract_hog_features() **اگر مقدار** pixels_per_cell **را عدد کوچک‌تری مثل 4 به جای 6 بگذاریم تصویر خروجی واضح‌تر خواهد شد.(دقت بیش‌تری خواهد داشت)**



عکس 2 : اعمال فیلتر HOG بر روی تصویر و تاثیر آن

3- **فیلتر دلخواه**

فیلتر Sharpen برای افزایش کنتراست و وضوح لبه‌ها استفاده می‌شود. این فیلتر باعث برجسته‌تر شدن ویژگی‌های اصلی اعداد می‌شود، اما ممکن است نویز را هم تقویت کند. برای بهبود این فیلتر می‌توان مقدار kernel **را در** filter2D() **تغییر داد. برای شارپ‌تر کردن تصویر می‌توان مقدار آن را افزایش داد.**



عکس 3: اعمال فیلتر شارپ بر روی تصویر و تاثیر آن

4- فیلتر sobel + hog

ترکیب ویژگی‌های Sobel و HOG ، توانایی توصیف ویژگی‌ها را افزایش می‌دهد. از آن جایی که Sobel لبه‌ها و HOG بافت و هندسه را درمی‌آورد، این ترکیب می‌تواند برای تصاویر با تفاوت‌های ظریف موثرتر باشد. برای مدل‌هایی که نیاز به ترکیب ویژگی‌های ساده و پیشرفته دارند، موثر است و دقت را در طبقه‌بندی نهایی افزایش می‌دهد.

5- فیلتر دلخواه + hog

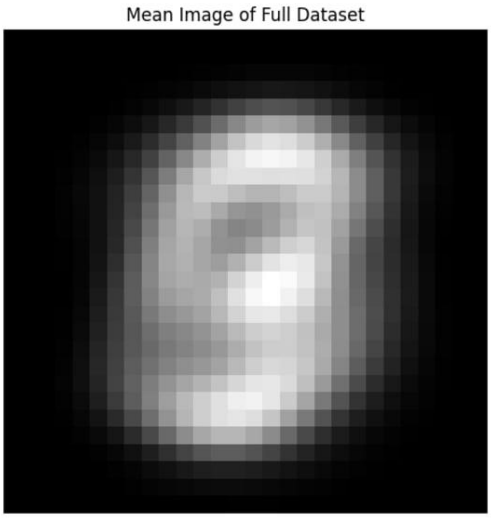
این فیلتر به دلیل استفاده از ویژگی‌های دقیق HOG به همراه تصویر پردازش‌شده توسط فیلتر شارپ، ویژگی‌هایی با کیفیت بالاتر ارائه می‌دهد. این فیلتر تاثیر بیش‌تری در شناسایی ویژگی‌های ساختاری و کاهش نویز دارد و همچنین برای اعداد دست‌نویس با نویز بیشتر یا دقت پایین‌تر می‌تواند کارآمدتر باشد. از آن‌جایی که ویژگی‌های دقیق‌تر و نویز کمتر باعث بهبود فرآیند یادگیری می‌شود می‌توان گفت این فیلتر عملکرد بهتری نسبت به بقیه فیلترها دارد.

6- داده خام

پردازش تصاویر خام بدون پیدا کردن ویژگی‌ها می‌تواند باعث کاهش دقت در دسته‌بندی نهایی شود، زیرا تفاوت‌های ظریف بین کلاس‌ها در فضای خام به‌خوبی قابل تفکیک نیستند. تاثیر داده‌خام در طبقه‌بندی به این صورت است که برای مدلی مثل SVM ممکن است عملکرد ضعیفی داشته باشند زیرا داده بدون فیلتر نویز بیشتری دارد. استفاده از پیش‌پردازش‌های مناسب برای کاهش نویز و تاکید بر ویژگی‌های اصلی در داده خام می‌تواند به بهبود آن کمک کند.

فاز دوم: Image Centering and PCA

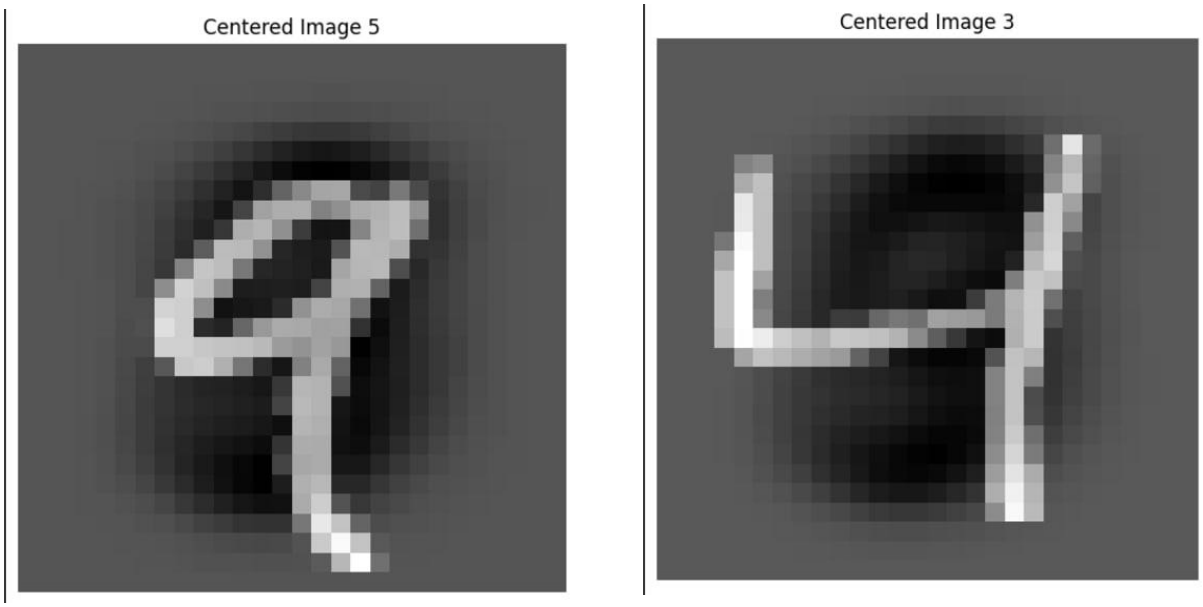
ابتدا در این فاز میانگین تمام تصاویر دیتاست را با استفاده از تابع calculate_mean_image() **به دست می‌آوریم.**



عکس 4 : میانگین تمامی تصاویر دیتاست

عکس 4 نشان می‌دهد که کدام نواحی در تصاویر بیشترین تأثیر را در میان تمام نمونه‌ها دارند.

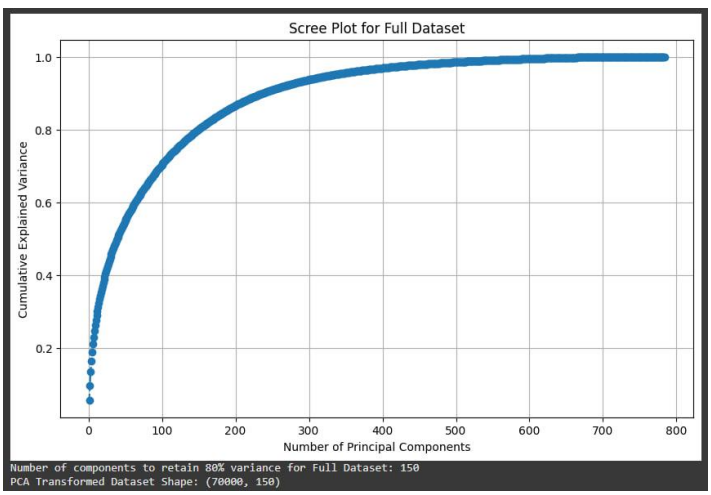
در ادامه به کمک تابع center_images() **میانگین به دست آمده را از هر تصویر کم می‌کنیم. با این کار، همه داده‌ها متمرکز می‌شوند و نویزهای عمومی در آن‌ها کاهش می‌یابد. این مرحله پیش‌نیازی برای کاهش ابعاد با** PCA **است.**

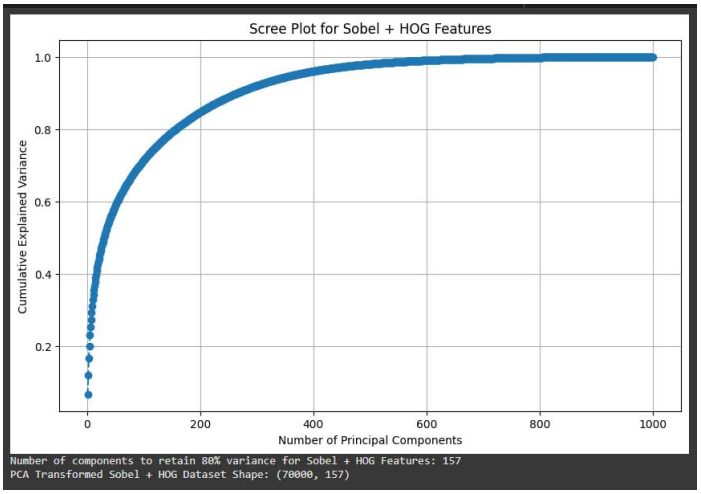


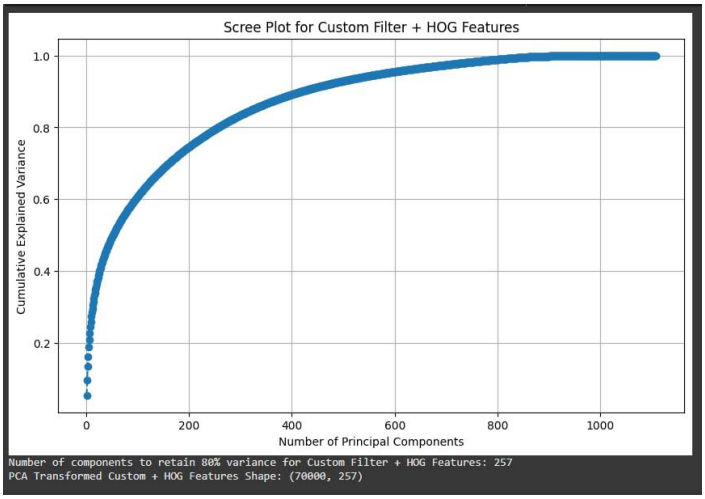
عکس 5 : تصاویر centerd

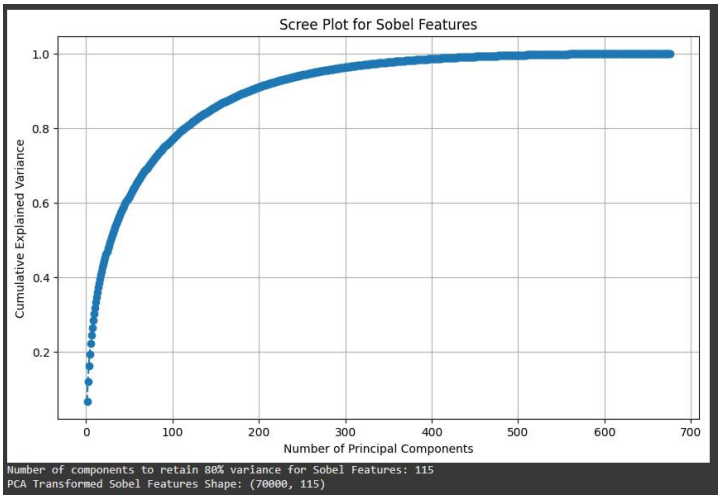
این تصاویر ظاهری مشابه تصاویر اصلی دارند اما ممکن است در برخی نواحی تیره‌تر و برخی نواحی روشن‌تر باشند.

در ادامه با کمک PCA به کاهش ابعاد داده می‌پردازیم. برای اینکه مشخص کنیم چند مولفه‌اصلی برای حفظ درصد مشخصی از واریانس داده نیاز هست، از scree plot کمک می‌گیرم. این نمودار نشان می‌دهد که با افزایش تعداد مؤلفه‌ها، چه مقدار از واریانس داده پوشش داده می‌شود. با مشخص کردن explained_variance >= 0.80 ، ما تعیین کردیم که 80 درصد واریانس داده پوشش داده شود. در واقع می‌توان گفت 80 درصد از تنوع موجود در داده‌ها توسط تعدادی مؤلفه اصلی انتخاب‌شده، حفظ شده است و یعنی تعداد مؤلفه‌های اصلی انتخاب‌شده، داده‌ها را به فضایی با ابعاد کمتر نگاشت داده که همچنان اطلاعات اصلی را حفظ کرده است.در زیر scree plot های رسم شده برای چهار مجموعه داده مختلف را می‌بینیم:









عکس 6 : محاسبه n-componant با کمک scree plot

برای هریک از مجموعه داده‌های بالا با کمک تابع centering_and_pca() و مقداری که از نمودار برای n-componant به دست آورده‌ایم ، PCA را اجرا کرده و مجموعه داده‌ای با ابعاد کمتر به دست می‌آوریم.

در پایان این بخش با کمک تابع train_test_split() دیتاست را به دو بخش train و test و نسبت 80–20 تقسیم می‌کنیم.

فاز سوم: Decision Tree Hyperparameter Tuning

در این فاز ما به دنبال یافتن بهینه‌سازی هایپرپارامترهای درخت تصمیم و در ادامه آن مقایسه عملکرد آن با یک مدل SVM هستیم.

برای پیدا کردن هایپرپارامترهای بهینه، از Grid Search استفاده کردیم. همچنین درحین استفاده از Grid Search از 5-Fold Cross-Validation برای ارزیابی مدل خود استفاده کردیم که باعث معتبرتر شدن نتایج ما می‌شود.

خروجی grid_search.best_params_ بهترین مقادیر هایپرپارامترها را مشخص می‌کند. باید دقت داشت که مقادیری مثل max_depth نباید خیلی زیاد باشد وگرنه منجربه overfitting می‌شود.
برای جلوگیری از ایجاد گره‌های بسیار کوچک یا تقسیمات غیرمنطقی نیز باید مقادیر min_samples_leaf و min_samples_split معقول و منطقی باشند.

نتایج اجرا Grid Search به صورت زیر است:

Raw Data

```
1 best_params_1, val_accuracy_1, best_model_1 = perform_grid_search( x_train_1 , x_test_1 , y_train_1 , y_test_1)
2 print( best_params_1, val_accuracy_1 )
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
Best Parameters: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5}
Validation Accuracy: 0.82
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5} 0.8177857142857143
```

Combined Sobel HOG

```
1 best_params_2, val_accuracy_2, best_model_2 = perform_grid_search( x_train_2 , x_test_2 , y_train_2 , y_test_2)
2 print( best_params_2, val_accuracy_2)
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
Best Parameters: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 5, 'min_samples_split': 2}
Validation Accuracy: 0.81
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 5, 'min_samples_split': 2} 0.8084285714285714
```

Sobel

```
1 best_params_3, val_accuracy_3, best_model_3 = perform_grid_search( x_train_3 , x_test_3 , y_train_3 , y_test_3)
2 print( best_params_3, val_accuracy_3)
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
Best Parameters: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5}
Validation Accuracy: 0.80
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5} 0.8023571428571429
```

Combined Custom HOG

```
1 best_params_4, val_accuracy_4, best_model_4 = perform_grid_search( x_train_4 , x_test_4 , y_train_4 , y_test_4)
2 print( best_params_4, val_accuracy_4)
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
Best Parameters: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 5, 'min_samples_split': 2}
Validation Accuracy: 0.81
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 5, 'min_samples_split': 2} 0.8119285714285714
```

نتایج به‌دست‌آمده برای دقت مدل درخت تصمیم به شرح زیر است :

Sobel - Decision Tree Accuracy: 0.802

Raw Data - Decision Tree Accuracy: 0.818

Combined Sobel HOG - Decision Tree Accuracy: 0.808

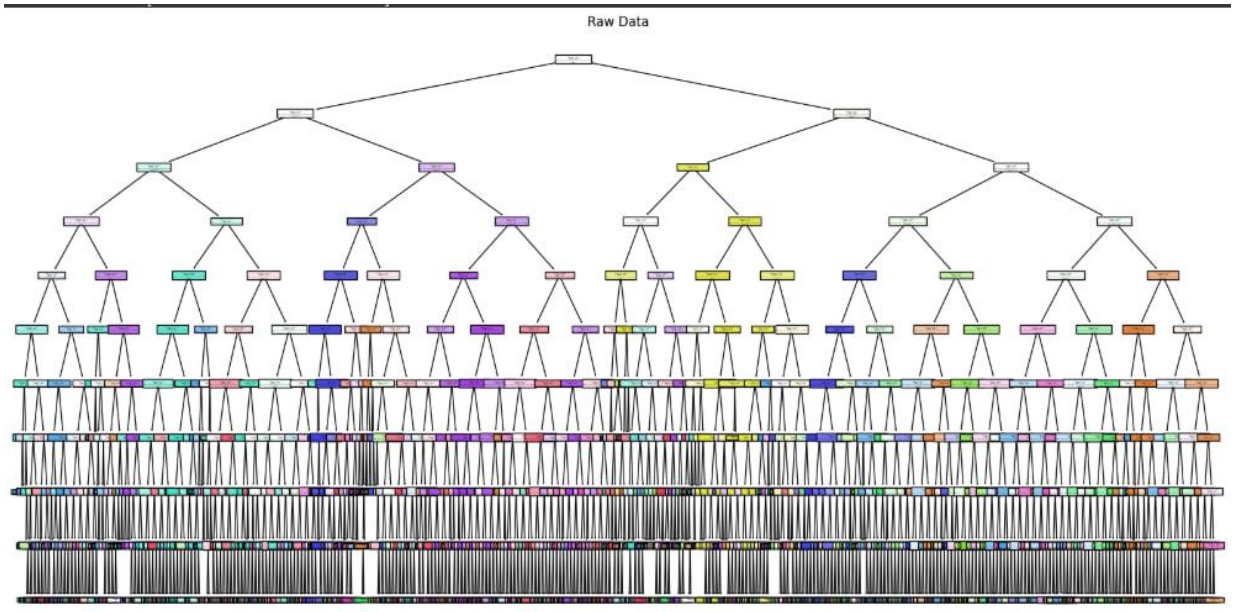
Combined Custom HOG - Decision Tree Accuracy: 0.812

همانطور که از نتایج مشخص است داده‌خام عملکرد بهتری نسبت به بقیه دارد و ازآن‌جایی که داده خام شامل تمام اطلاعات پیکسل‌ها بدون تبدیل می‌باشد، این نتیجه منطقی است.

از آنجایی که داده‌های Combined ویژگی‌های خاصی مثل لبه‌ها را به‌صورت چکیده‌تر نمایش می‌دهند و ممکن است برخی اطلاعات از دست برود، مقداری عملکردشان ضعیف‌تر از داده خام است.

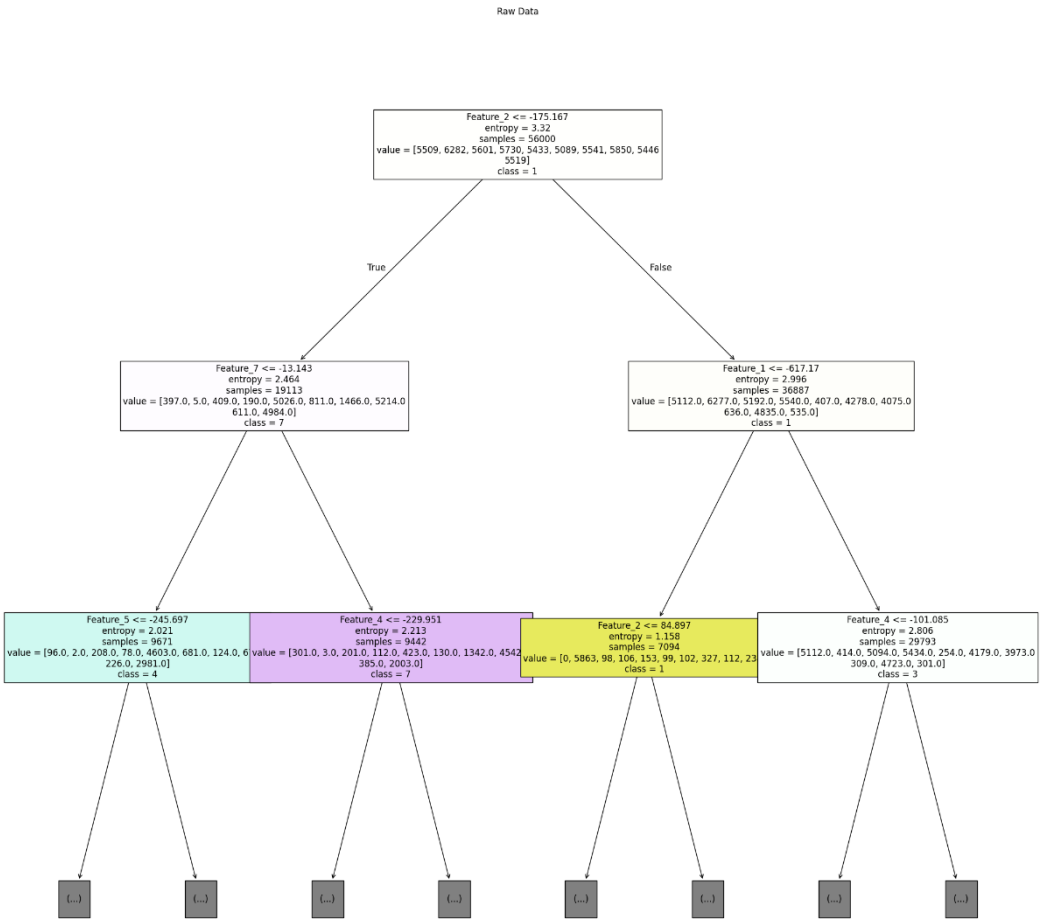
همچنین sobel ضعیف‌ترین نتیجه را نسبت به همه دارد که دلیل آن می‌تواند ساده‌تر بودن ویژگی‌ها (فقط لبه‌ها) و کاهش زیاد ابعاد باشد.

درخت داده‌های خام به صورت زیر است:



عکس 7 : درخت تصمیم داده‌های خام

برای تحلیل و بررسی بهتر درخت را با max_depth =2 رسم می‌کنیم:



از آنجایی که feature2 به عنوان جدا کننده در در گره ریشه انتخاب شده مهم ترین ویژگی مدل شناسایی شده است.این نشان می‌دهد که این ویژگی ممکن است با خصوصیت غالب در داده‌ها ارتباط نزدیکی داشته باشد همچنین ممکن است با یک ویژگی ساختاری یا عددی خاص مرتبط باشد که تأثیر مستقیم بر طبقه‌بندی دارد.در سطح بعدی شاخه سمت چپ دیتاهایی قرار گرفتند که مقدار feature2 آنها پایین است و بیشتر مرتبط به کلاس 7 هستند. در این گره می‌بینیم که مقدار انتروپی کاهش یافته که این نشان دهنده این است که مدل در جداسازی داده ها پیشرفت داشته است.

در گره سمت راست مقدار feature2 بیش‌تر بوده و مقادیر کوچک feature1 بیش‌تر به کلاس 1 مرتبط هستند.از طرفی در این گره مقدار انتروپی بالاتر رفته و این نشان می‌دهد داده ها در این شاخه تنوع بیش‌تری دارند و مدل هنوز نتوانسته به خوبی آنها را جدا کند.

گره‌هایی با انتروپی بالا نشان‌دهنده احتمال اشتباه طبقه‌بندی هستند.

مثلا در مسیر گره با Feature 4 <= -101.085، مدل نتوانسته است بین کلاس‌های 3 و 1 تمایز کاملی ایجاد کند.

نتایج به دست آمده برای مدل SVM نیز به صورت زیر می‌باشد:

Raw Data:
Training Accuracy: 0.9935892857142857
SVM Accuracy: 0.9814285714285714
Combined Sobel HOG:
Training Accuracy: 0.98925
SVM Accuracy: 0.9712142857142857

همانطور که ازنتایج مشخص است مدل SVM برای هردو حالت، درآموزش دقت بالایی دارد.
همانند دقت آموزش، دقت تست هم بالاست که این نتیجه نشان‌دهنده عملکرد خیلی خوب SVM است. دقت بالای تست به ما نشان می‌دهد که مدل به خوبی تعمیم یافته است و مشکل overfitting وجود ندارد.
همانند درخت تصمیم در مدل SVM هم نتیجه داده‌خام بهتر است که دلیل آن به خاطر اطلاعات بیش‌تری است که داده‌خام دارد.

با مشاهده و مقایسه نتایج درخت تصمیم و مدل SVM متوجه می‌شویم، مدل SVM عملکرد بهتری نسبت به درخت تصمیم از خود نشان می‌دهد(مخصوصا برای داده خام)
علت نتایج بهتر مدل SVM نسبت به درخت تصمیم را می‌توان عواملی همچون: عملکرد بهتر SVM برای داده‌هایی با ابعاد بالا و غیر خطی و حساسیت درخت تصمیم نسبت به تنظیم دقیق هایپرپارامترها و آسیب‌پذیری نسبت به overfitting دانست.

به طور کلی می‌توان گفت برای دیتاست MNIST که تصاویر ساختارهای بسیار پیچیده‌ای ندارند، انجام عملیات‌ها و فیلترهای مختلف مثل sobel , hog ممکن است جزئیات پیکسل ها را حذف کنند ویا ویژگی‌های اضافه‌ای را اضافه کنند و بر نتیجه نهایی ما تاثیر بگذارند.به همین دلیل نتایجی که از دیتای خام به دست آوردیم دقت بالاتری دارند و نتایج بهتری را به ما نشان می‌دهد.

فاز چهارم: تحلیل دقت مدل و معیارهای سنجش

در این فاز به تحلیل دقیق عملکرد مدل می‌پردازیم. برای هر کلاس معیارهای Precision، Recall و F1-Score را با کمک تابع evaluate_model_performance() به دست میاوریم. نتیجه اجرا این تابع برای درخت تصمیم به صورت زیر است:

Performance Report for Raw Data:

	precision	recall	f1-score	support
0	0.91	0.88	0.90	1314
1	0.95	0.97	0.96	1584
2	0.81	0.87	0.84	1392
3	0.77	0.79	0.78	1450
4	0.90	0.81	0.85	1374
5	0.77	0.77	0.77	1266
6	0.92	0.91	0.91	1386
7	0.89	0.90	0.89	1484
8	0.77	0.73	0.75	1333
9	0.79	0.83	0.81	1417
accuracy			0.85	14000
macro avg	0.85	0.85	0.85	14000
weighted avg	0.85	0.85	0.85	14000

Performance Report for Combined Sobel HOG:

	precision	recall	f1-score	support
0	0.86	0.91	0.88	1406
1	0.97	0.97	0.97	1611
2	0.83	0.86	0.84	1332
3	0.77	0.79	0.78	1402
4	0.82	0.78	0.80	1388
5	0.85	0.69	0.76	1290
6	0.92	0.90	0.91	1409
7	0.85	0.85	0.85	1456
8	0.77	0.79	0.78	1354
9	0.77	0.82	0.80	1352
accuracy			0.84	14000
macro avg	0.84	0.84	0.84	14000
weighted avg	0.84	0.84	0.84	14000

Performance Report for Sobel:

	precision	recall	f1-score	support
0	0.85	0.92	0.88	1349
1	0.98	0.96	0.97	1600
2	0.86	0.87	0.86	1374
3	0.78	0.80	0.79	1441
4	0.82	0.81	0.82	1384
5	0.82	0.73	0.77	1264
6	0.94	0.91	0.93	1361
7	0.85	0.83	0.84	1447
8	0.79	0.79	0.79	1400
9	0.75	0.79	0.77	1380
accuracy			0.84	14000
macro avg	0.84	0.84	0.84	14000
weighted avg	0.85	0.84	0.84	14000

Performance Report for Combined Custom HOG:

	precision	recall	f1-score	support
0	0.93	0.89	0.91	1333
1	0.97	0.96	0.97	1564
2	0.80	0.82	0.81	1387
3	0.77	0.77	0.77	1435
4	0.84	0.82	0.83	1376
5	0.75	0.80	0.77	1292
6	0.91	0.91	0.91	1407
7	0.88	0.89	0.89	1492
8	0.77	0.77	0.77	1333
9	0.82	0.80	0.81	1381
accuracy			0.85	14000
macro avg	0.85	0.84	0.84	14000
weighted avg	0.85	0.85	0.85	14000

با نگاه به خروجی‌ها به نتایج زیر می‌رسیم:

داده‌های خام:

Precision و Recall حدود 0.85 است. و کلاس‌های 3 و 5 و 8 Precision و Recall کمتری دارند پس این کلاس‌ها به‌احتمال زیاد شباهت بیشتری به کلاس‌های دیگر دارند و یا اینکه الگوهای آن‌ها به‌خوبی توسط ویژگی‌های درخت شناسایی نمی‌شود.

F1-Score کلاس‌ها نشان دهنده این است که عملکرد درخت تصمیم در تشخیص اکثر کلاس‌ها متعادل است، اما با این حال در برخی کلاس‌ها می‌تواند بهتر شود و بهبود نیاز دارد.

Combined Sobel HOGو Sobel :

عملکرد کلی آن‌ها مثل داده‌های خام است اما Precision و Recall در بعضی کلاس‌ها کمتر شده است و این موضوع هم به دلیل کاهش اطلاعات زمان به دست آوردن ویژگی‌ها است.

یکی از نقاط ضعف درخت تصمیم حساسیت به رابطه‌های خطی و ساده بین ویژگی‌هاست و به همین دلیل ممکن هست ارتباطات پیچیده بین پیکسل‌های دیتاست ما را نتواند به‌خوبی مدل کند. همانطور که در فاز قبلی هم گفته شد یکی از نقاط ضعف درخت تصمیم می‌تواند overfitting **باشد که دقت بالای آموزش در فاز قبلی نشان‌دهنده تمرکز بیش‌از حد مدل درخت تصمیم روی داده‌های آموزشی است که می‌توان این موضوع را روی داده تست و** recall **پایین در کلاس‌های خاص دید.**

نتیجه اجرا این تابع برای SVM **نیز به صورت زیر است:**

Performance Report for Raw Data:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	1314
1	0.99	0.99	0.99	1584
2	0.99	0.99	0.99	1392
3	0.99	0.99	0.99	1450
4	0.99	0.99	0.99	1374
5	1.00	0.99	0.99	1266
6	0.99	1.00	1.00	1386
7	0.99	0.99	0.99	1484
8	0.99	0.99	0.99	1333
9	0.98	0.98	0.98	1417
accuracy			0.99	14000
macro avg	0.99	0.99	0.99	14000
weighted avg	0.99	0.99	0.99	14000

Performance Report for Combined Sobel HOG:

	precision	recall	f1-score	support
0	0.99	1.00	0.99	1406
1	0.99	1.00	1.00	1611
2	0.99	0.99	0.99	1332
3	0.98	0.98	0.98	1402
4	0.98	0.98	0.98	1388
5	0.99	0.98	0.99	1290
6	0.99	0.99	0.99	1409
7	0.99	0.99	0.99	1456
8	0.98	0.99	0.98	1354
9	0.97	0.98	0.98	1352
accuracy			0.99	14000
macro avg	0.99	0.99	0.99	14000
weighted avg	0.99	0.99	0.99	14000

با نگاه به خروجی‌ها به نتایج زیر می‌رسیم:

داده‌های خام:

عملکرد نزدیک به ایده‌آلی داریم و میانگین دقت و Recall **و** F1-Score **حدودا 0.99 هست و همه کلاس‌ها با دقت بالا شناسایی شده‌اند.**

دقت مدل نشان دهنده این است که SVM **دیتاست ما را به‌خوبی مدل کرده است.**

Combined Sobel HOG:

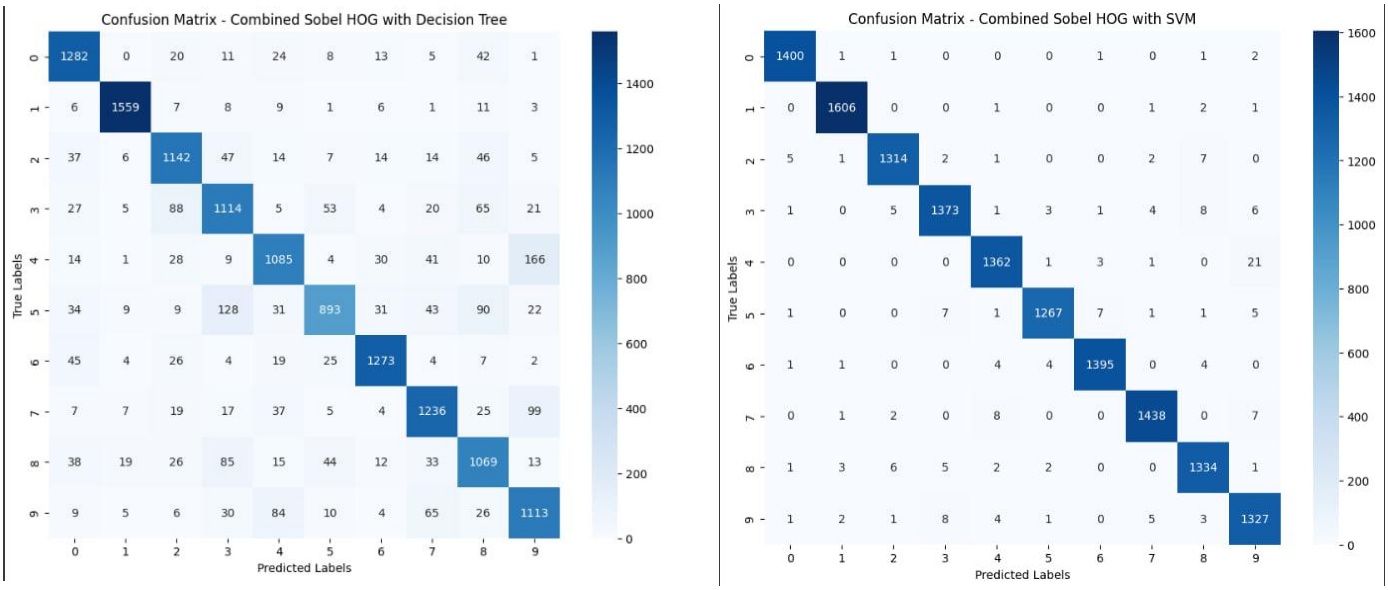
عملکرد همچنان خیلی بالاست. به دست آوردن ویژگی‌ها (Sobel + HOG) باعث کاهش بسیار کمی در اطلاعات شده، اما به طور کلی تاثیری در عملکرد کلی نداشته است.

نقطه قوت مدل SVM **این است که برای کرنل‌های غیرخطی نتیجه خیلی خوبی را به ما می‌دهد. یکی دیگر از ویژگی‌های خوب مدل** SVM **این است که حتی در داده‌های کاهش‌یافته (Sobel + HOG) هم عملکرد بسیار بالایی نشان می‌دهد.**

Precision و Recall و F1-Score به طور واضحی از درخت تصمیم بالاتر هست و از آن‌جا که SVM بهتر می‌تواند مرزهای پیچیده بین کلاس‌ها را مدل کند پس این نتیجه منطقی است.

برای اینکه اطلاعات بیش‌تری را درمورد عملکرد مدل‌ها به دست بیاوریم از Confusion Matrices استفاده می‌کنیم. با استفاده از تابع plot_confusion_matrix() ماتریس را به صورت heatmap ، نمایش می‌دهیم. این ماتریس، عملکرد یک مدل را با نشان دادن چگونگی تطابق لیبل‌های واقعی با لیبل‌های پیش‌بینی‌شده، نشان می‌دهد.

خروجی heatmap درخت تصمیم و SVM ما به صورت زیر است:



اعدادی که روی قطر ماتریس قرار دارند پیش‌بینی‌های درست هستند.

اعدادی که روی قطر قرار ندارند نشان دهنده اشتباهات کلاس دسته‌بندی هستند.

با مقایسه بین مدل‌ها متوجه می‌شویم در مدل درخت تصمیم اشتباهات دسته‌بندی بیش‌تری نسبت به مدل SVM وجود دارد.(اعداد غیرقطری بیش‌تری در مدل درخت تصمیم هست که مقدار بزرگی داشته باشند.)

مدل SVM عملکرد بهتری دارد چون بیشتر عناصر غیرقطری 0 یا بسیار کم هستند، یعنی اشتباهات دسته‌بندی کمتری وجود دارد.

طبق خروجی که می‌بینیم در مدل درخت تصمیم، کلاس‌های(4 و 9) و (5 و 3) اغلب باهم اشتباه گرفته می‌شوند. برای این کلاس‌ها یکی از دلایل را می‌توان شباهت ویژگی‌های بصری آن‌ها باشد.

فاز پنجم: Overfitting and Pruning

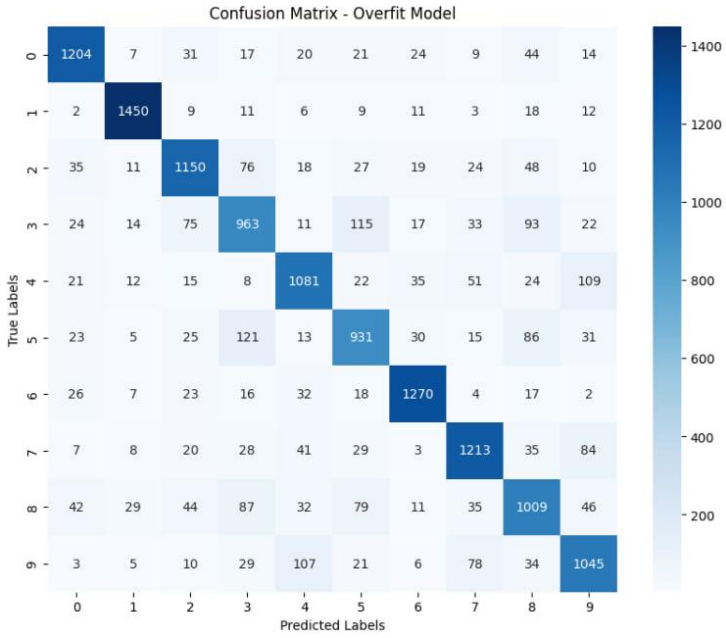
در این فاز ابتدا سعی می‌کنیم با انجام اقداماتی روی مدل باعث overfit آن شویم. برای این کار باید محدودیت‌هایی که برای درخت تصمیم تنظیم می‌شود را برداریم و کاری کنیم که بدون کنترل روی داده‌های آموزش عمل کند.
برای این کار می‌توانیم عمق درخت را محدود نکنیم پس max_depth را None مقداردهی می‌کنیم.
کار دیگه‌ای که می‌توان انجام داد این است که کمترین تعداد نمونه برای تقسیم (min_samples_split) را عددی بسیار کوچک انتخاب کنیم این کار باعث می‌شود که درخت حتی با تعداد کم نمونه‌ها به تقسیم داده‌ها ادامه بدهد.
همچنین اگر کمترین تعداد نمونه در برگ‌ها (min_samples_leaf) هم خیلی کوچک باشد به درخت اجازه می‌دهد هر نمونه جداگانه در برگ‌ها قرار گیرد.

با استفاده از تابع train_and_evaluate_decision_tree() و فراخوانی آن و مقداردهی به عنوان ورودی با شرایطی که بالا گفته شد، توقع می‌رود به علت overfit دقت آموزش ما خیلی بالا و 1 باشد، چون مدل کاملاً بر داده‌های آموزشی منطبق می‌شود. در حالی که چون مدل به دلیل overfitting روی داده‌های جدید عملکرد ضعیفی خواهد داشت، توقع داریم دقت تست کم‌تر از دقت آموزش باشد.
نتیجه ما به صورت زیر است:

Overfit Model - Train Accuracy: 1.00

Overfit Model - Test Accuracy: 0.81

خروجی که داریم مشابه توقع ما از مدلی است که overfit شده است. برای تحلیل بیش‌تر confusion_matrix آن را نیز رسم می‌کنیم:



عکس 8 : مدل overfit شده

ماتریس هم نشان می‌دهد که دقت روی قطر اصلی بسیار بالاست. این نشان‌دهنده این است که مدل داده‌های آموزشی را بسیار خوب یاد گرفته است اما ممکن است روی داده‌های جدید به خاطر overfitting عملکرد ضعیفی داشته باشد. همچنین مقادیر خارج از قطر اصلی نسبتاً کم هستند که این مورد با فرضیه overfitting هم‌خوانی دارد.

در مرحله بعدی این فاز، پس از overfit شدن مدل سعی می‌کنیم با استفاده از تکنیک pre-pruning و محدود کردن هایپرپارامترها، overfitting مدل را کاهش دهیم.

در این مرحله باید مقادیر منطقی برای هایپرپارامترهای خود (max_depth و min_samples_split و min_samples_leaf) انتخاب کنیم و بعد از آن مدل خود را با این تنظیمات آموزش دهیم.

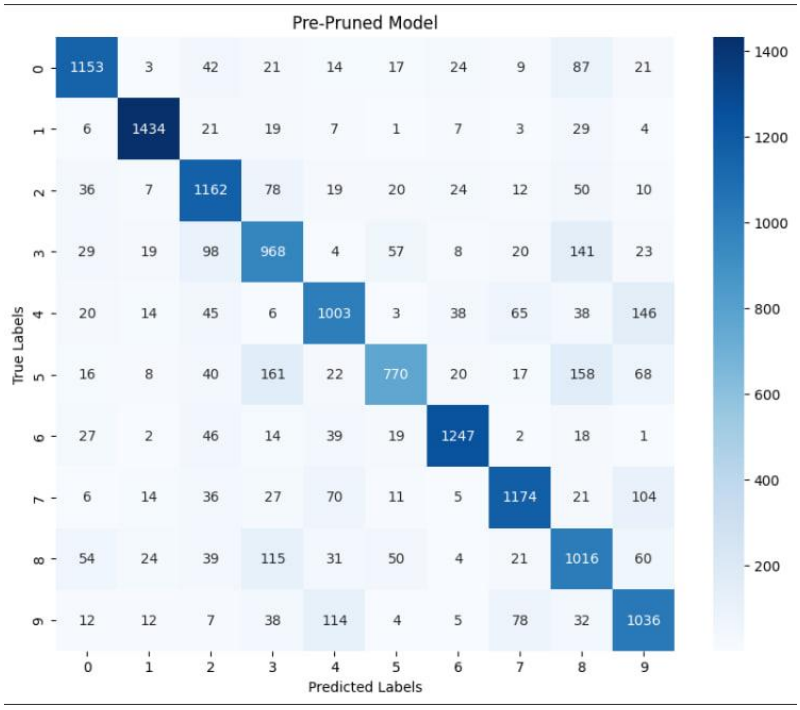
در این حالت توقع داریم دقت تست بهتر شود و یا حداقل نزدیک به دقت آموزش شود. عملا می‌توان گفت باید فاصله میان دقت آموزش و تست باید کم شود.

نتیجه ما به صورت زیر است:

Model - Train Accuracy: 0.81

Model - Test Accuracy: 0.78

خروجی که داریم مشابه توقع ما از مدلی است که overfit شده است. زیرا دقت فاصله دقت آموزشی و تست کم شده و دقت آموزش هم از مقدار 1 کم‌تر شده است. . برای تحلیل بیش‌تر confusion_matrix آن را نیز رسم می‌کنیم:



عکس 9 : مدل pre-pruned

همانطور که مشاهده می‌شود، این ماتریس نسبت به مدل قبلی کمتر متمرکز است و خطاها پراکنده‌تر شده‌اند، که نشانه‌ای از بهبود تعمیم‌دهی است. در مدل overfit **، خطاهای بسیار کمی در داده‌های آموزش وجود داشت، اما در داده‌های تست عملکرد مدل کاهش می‌یافت. در این حالت مدل ما خطاهای تقریبا متعادلی روی داده‌ها دارد.**

استفاده از تکنیک‌های pre-pruning **باعث می‌شود که مدل با داده‌های آموزش زیاد تطابق پیدا نکند و در عوض بهتر روی داده‌های تست تعمیم پیدا کند.**

نتیجه می‌گیریم: pre-pruning **به درستی** overfitting **را کاهش داده است.**