پروژه پایانی یادگیری ماشین

شایان بمانیان

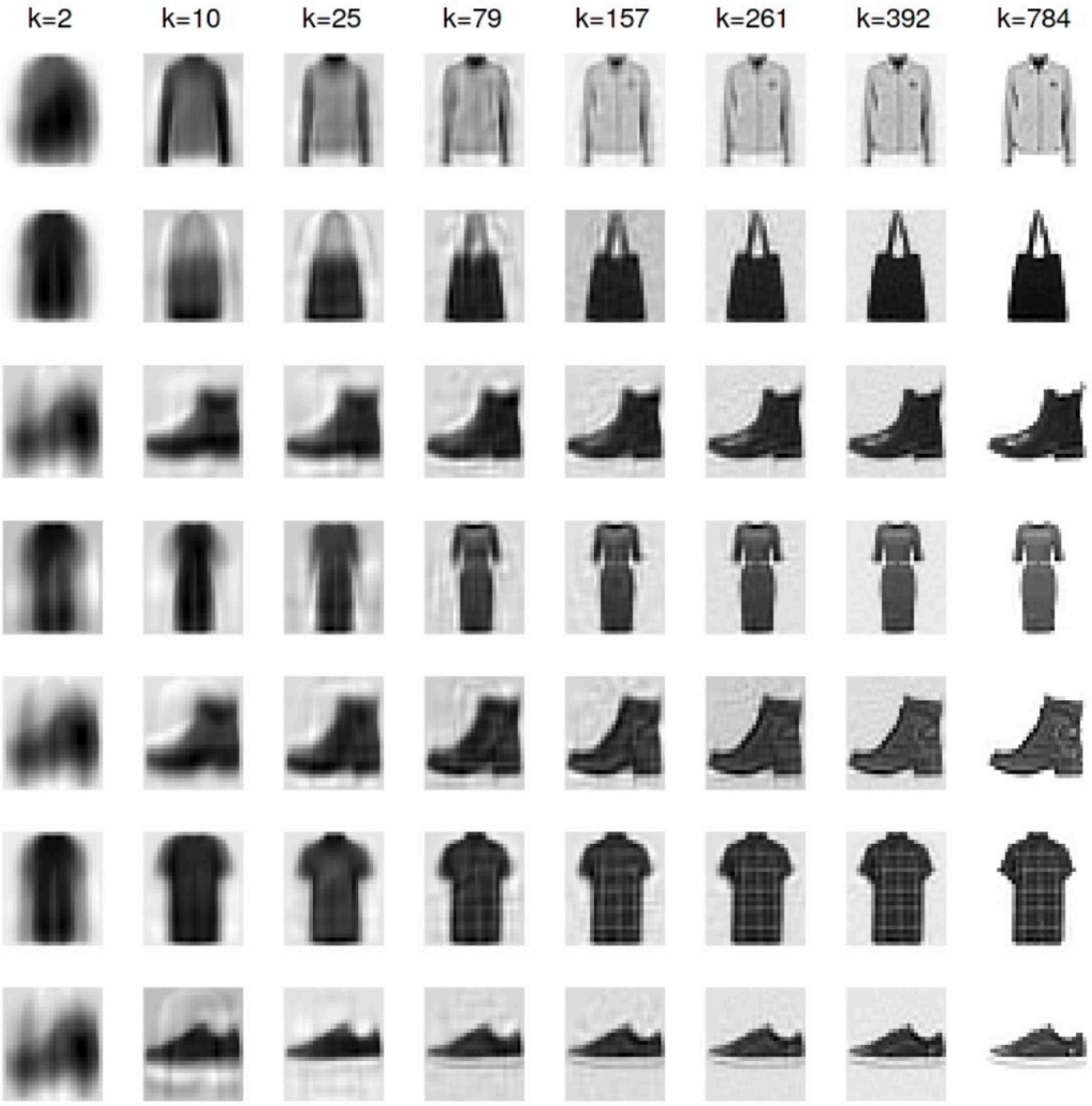
مقدمه

مدل مخفی مارکوف یک روش محبوب برای مدل‌سازی داده‌های زمانی و سیکونشال است. این مدل از یک سری وضعیت یا حالت‌های نهان، که تعیین‌کننده فرایند هستند تشکیل شده و با استفاده از پارامترهایی که از داده‌های آموزشی به‌دست می‌آید، احتمال ورود به هر یک از این حالت‌ها در زمان‌های مختلف، محاسبه می‌شود. در این پروژه ما قصد داریم از این مدل برای انجام دسته‌بندی روی داده دیتاست FashionMNIST استفاده کنیم. هرچند این مدل جزو دسته‌بند های معروف شناخته نمی‌شود اما ما با استفاده از استخراج ویژگی های درست و تنظیم هایپرپارامتر ها به دقت ۸۷٪ روی این دیتاست دست پیدا کنیم. چالش های اصلی این پروژه شامل پیدا کردن هایپر پارامتر های مناسب و معماری درست برای مدل مارکوف و استخراج درست ویژگی ها از تصویر بود.

توضیح روال پیاده‌سازی

با توجه به inductive bias یا گرایش استنتاجی مدل مخفی مارکوف به داده های زمانی و متوالی، ما باید تصویر خود را به صورت سیکونشال درآوریم. برای این کار در تلاش اول تصویر استاندارد ۲۸ در ۲۸ پیکسلی MNIST را فلت کرده که حاصل آن ۷۸۴ پیکسل می‌شود و می‌توان آن را به صورت متوالی به مدل داد. البته که با این روش مدل دقت ۲٪ را کسب می‌کرد. سپس از تصمیم گرفتیم از روش کاهش بعد با استفاده از PCA مدل را آموزش دهیم که دقت مدل خیلی تغییری نکرد. پس از جستجو زیاد به این نتیجه رسیدیم که بهتر است برای هر کلاس یک مدل جداگانه ترین کنیم. با این کار و با استفاده از روش کاهش بعد PCA به دقت ۶۷٪ رسیدیم. که با تغییر تعداد استیت های hmm و ابعاد خروجی PCA تا این عدد به ۷۴٪ رسید.

در تصویر زیر تاثیر PCA روی عکس های دیتاست ما آورده شده:



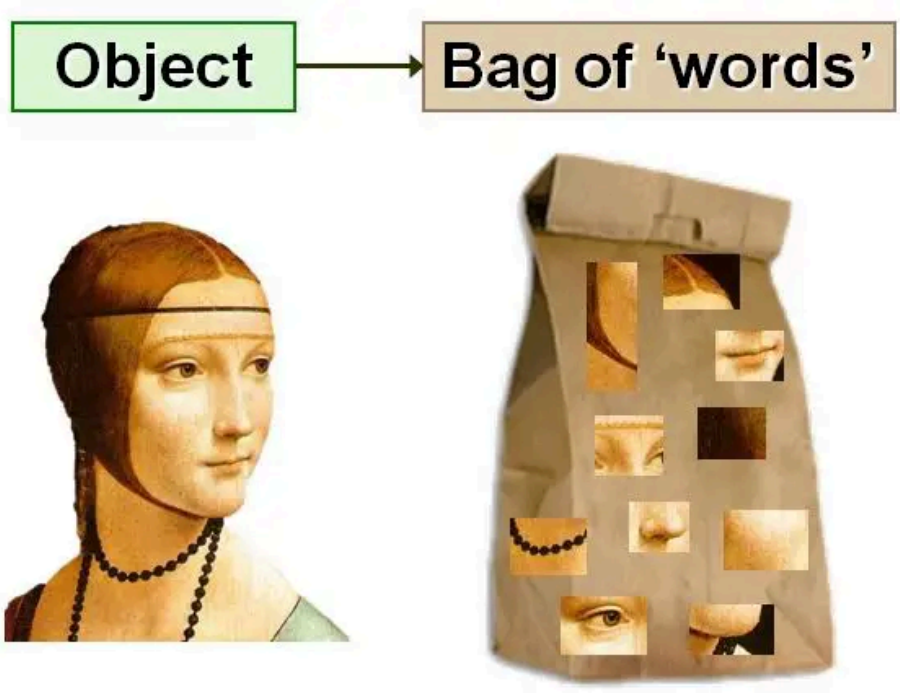
سپس تصمیم گرفتیم به سراغ روش های کاهش بعد مرسوم مثل LDA برویم. در LDA تعداد فیچر های خروجی آن به این صورت محاسبه می‌شود: min(n_classes - 1, n_features) در اینجا ۹ بود. چون تعداد فیچر ها کم بود تعداد استیت هارا نیز کاهش دادیم و با ۲ استیت به دقت ۸۰٪ رسیدیم. اما هرچقدر هایپرپارامتر هارا تغییر دادیم دقت ما از این عدد بالاتر نرفت.

ترکیب دو روش PCA و LDA نیز کمکی به افزایش دقت نکرد و حتی باعث کاهش آن نیز شد. به‌طور مثال به کد زیر توجه کنید:

```python
# PCA to reduce dimensionality
pca = PCA(n_components=157)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

# LDA to get feature vectors
lda = LinearDiscriminantAnalysis(n_components=9)  # n_components must be min(n_classes - 1, n_features)
X_train = lda.fit_transform(X_train_pca, y_train)
X_test = lda.transform(X_test_pca)

# One HMM per class
models = []
n_states = 25
for c in range(10):
    # Train HMM for each class
    model = hmm.GaussianHMM(n_components=n_states, covariance_type="tied")
    model.fit(X_train[y_train==c])
    models.append(model)

y_pred = []
for i in range(len(X_test)):
    obs = X_test[i]
    obs = obs.reshape(1, -1)  # Reshape to 2D
    scores = [model.score(obs) for model in models]
    y_pred.append(np.argmax(scores))
```

	precision	recall	f1-score	support
T-shirt/top	0.82	0.74	0.78	1000
Trouser	0.99	0.94	0.96	1000
Pullover	0.75	0.56	0.65	1000
Dress	0.66	0.74	0.70	1000
Coat	0.39	0.45	0.42	1000
Sandal	0.87	0.91	0.89	1000
Shirt	0.46	0.51	0.48	1000
Sneaker	0.91	0.85	0.88	1000
Bag	0.92	0.94	0.93	1000
Ankle boot	0.88	0.92	0.90	1000
accuracy			0.76	10000
macro avg	0.77	0.76	0.76	10000
weighted avg	0.77	0.76	0.76	10000

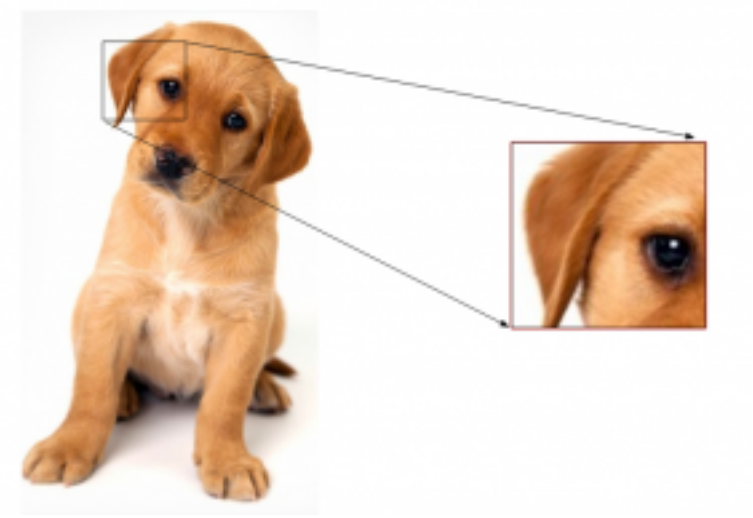
با جستجوی بیشتر و مراجعه به این مقاله به روشی به نام Visual bag of words رسیدیم. پایه Visual bag of words الگوریتم SIFT است. این الگوریتم نقاط کلیدی را شناسایی می‌کند و یک بردار از فیچر های اصلی تصویر به ما می‌دهد. این روش از متد TF-IDF استفاده می‌کند و به ما کمک می‌کند ویژگی خاص هر تصویر را به‌دست آوریم البته این الگوریتم برای دیتاست ما اصلا دقت مناسبی را ارائه نداد.



سپس به کتابخانه scikit-image رسیدیم که شامل چندین روش استخراج ویژگی از تصویر است. چند الگوریتم مرسوم استخراج ویژگی از تصویر شامل HOG، SIFT، LBP و Gabor Filter و یا Canny Edge Detector هستند.

اکثر مقاله هایی که در زمینه دسته‌بندی تصویر با استفاده از روش های کلاسیک ماشین لرنینگ ارائه شده بود از الگوریتم های HOG یا SIFT یا LBP یا SVD استفاده کرده بودند. تمامی روش ها را با هایپرپارامتر های مختلف امتحان کردیم ولی درنهایت بهترین دقت را با استفاده از الگوریتم HOG بدست آوردیم. توضیح الگوریتم **Histogram of Oriented Gradients** در این لینک به خوبی توضیح داده شده است. اما توضیح آن را به طور خلاصه در ادامه می‌آوریم.

این الگوریتم از گرادیان استفاده می‌کند. این تصویر را درنظر بگیرید:



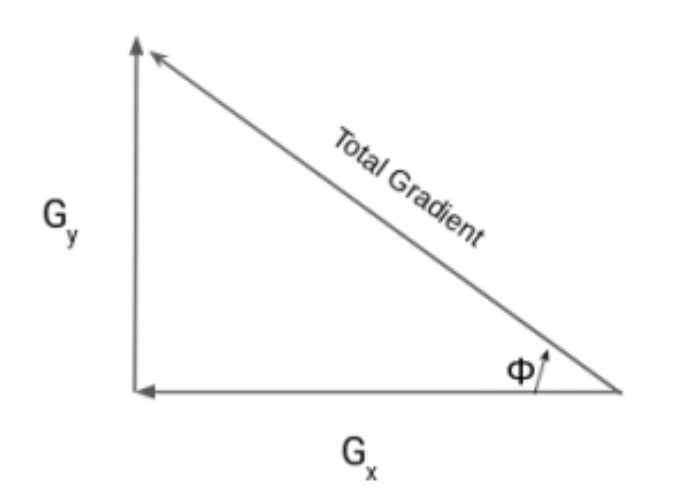
فرض می‌کنیم مقادیر پیکسلی قسمت زوم شده به‌صورت زیر باشد:

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

فرض کنید میخوایم گرادیان عدد ۸۵ را حساب کنیم. برای محاسبه تغییرات در راستای محور x ها ۸۵ را منهای عدد سمت چپ آن یعنی ۷۸ میکنیم و آن را Gx می‌نامیم. و برای محاسبه تغییرات در محور y ها عدد ۸۵ را منهای عدد پایین آن، یعنی ۵۶ می‌کنیم و آن را Gy می‌نامیم. با این کار الان دو متریک جدید داریم و این کار را برای تمام پیکسل های موجود در عکس تکرار می‌کنیم.

برای محاسبه اندازه و جهت گرادیان هر پیکسل با استفاده از گرادیان به‌دست آمده در مرحله قبل به‌صورت زیر عمل می‌کنیم.

برای محاسبه اندازه گرادیان:



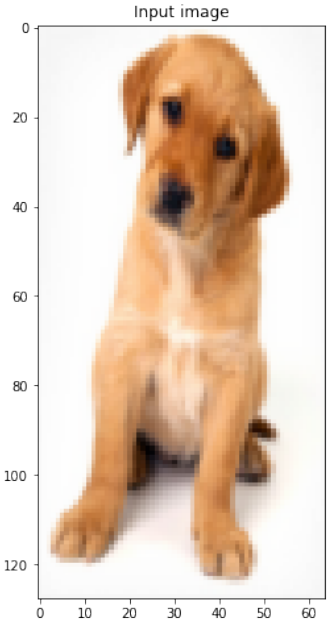
$$\text{Total Gradient Magnitude} = \sqrt{[(G_x)^2+(G_y)^2]}$$

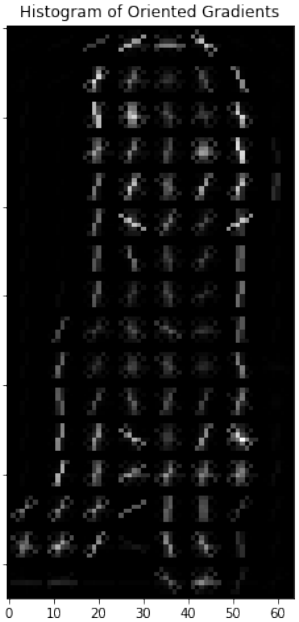
$$\text{Total Gradient Magnitude} = \sqrt{[(11)^2+(8)^2]} = 13.6$$

و برای محاسبه جهت آن به صورت زیر عمل می‌کنیم:

$$\tan(\Phi) = Gy / Gx$$

پس ما تا اینجا اندازه گرادیان هرپیکسل و جهت آن را داریم. در آخرین مرحله یک هیستوگرام می‌سازیم. هیستوگرام ها نشان دهنده تعداد تکرار جهت های مختلف را نشان می‌دهد. و در نهایت تصویر ما به صورت زیر درمی‌آید:





این مقاله روی دیتاست FashionMNIST با استفاده از کلاسیفایر SVM به دقت ۸۶.۵۳٪ رسیده بود

با توجه به قسمت زیر در مقاله، ما نیز هایپرپارامتر های خود را بر همین اساس انتخاب کردیم.

B. Histogram of Oriented Gradients (HOG)

One of the simple and effective feature extraction methods is HOG feature descriptor. It is a fast and efficient feature descriptor in compare to the SIFT and LBP due to the simple computations, it has been also shown that HOG features are successful descriptor for detection. Mainly it is used for object detection in image processing and computer vision. Using HOG the shape and appearance of the image can be described. It divides the image into small cells like 4-by-4 which is used in this work and computes the edge directions. For improving the accuracy the histograms can be normalized.

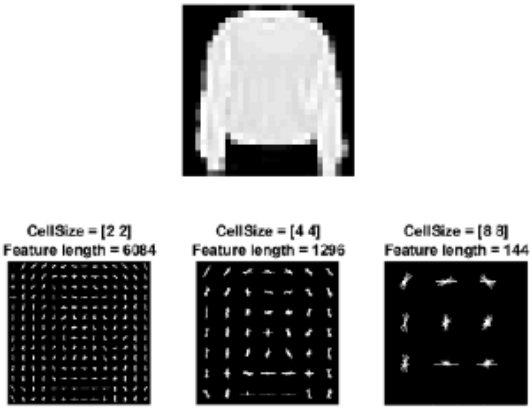
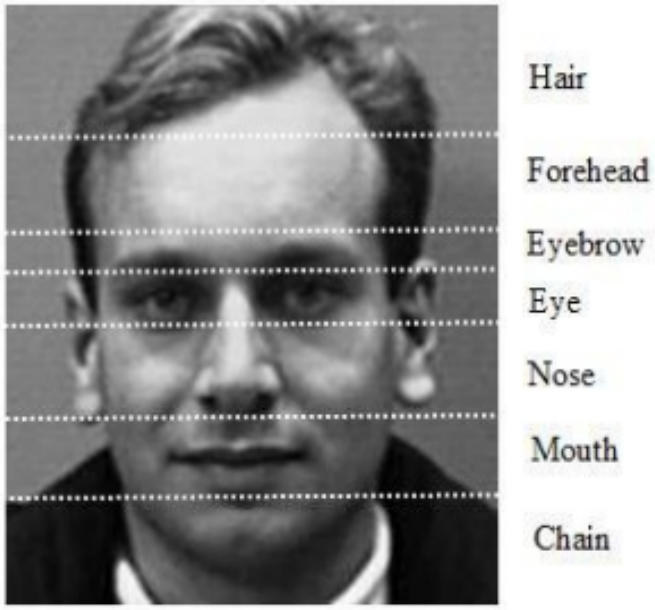


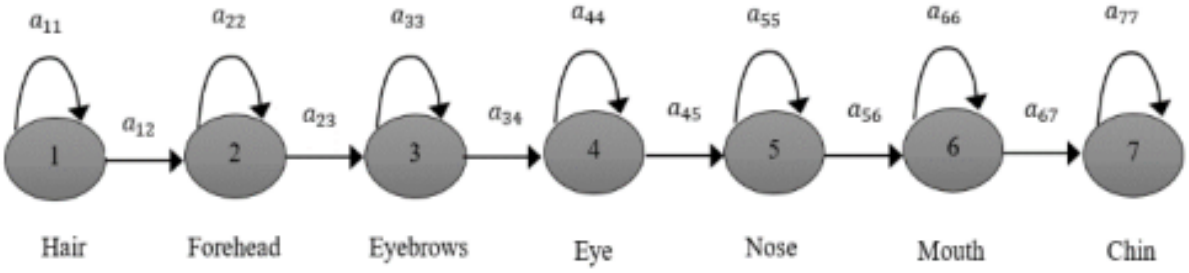
Fig. 2. Extracted features of an image in Fashion-MNIST Dataset

In Figure 2 extracted HOG features of one image using three different cell sizes are shown. In this figure the visualization of cell size [2 2], [4 4] and [8 8] are shown. From that it is clearly understood that the cell size [2 2] contains more shape information than the cell size of [8 8] in their visualization. But in the latter case the dimensionality of feature vector using HOG increases comparing with the former. A good choice is the [4 4] cell size. By using this size the numbers of dimensions are limited and this helps to speed up the training process. Also it contains enough information to visualize the fashion image shape. For identifying the suitable parameter setting configuration of HOG parameters more training and testing processes using the classifier has to be performed.

مقالات دیگری هم مثل این مقاله برای تشخیص چهره با استفاده از مدل مخفی مارکوف دیدیم که به ما برای تشخیص تعداد استیت های مدل کمک کرد.

در این مقاله آورده شده بود که برای تشخیص یک چهره، آن را به هفت قسمت شامل مو، پیشانی، ابرو، چشم، بینی، دهان و چانه تبدیل کرده و تعداد استیت های مدل مارکوف خود را بر این اساس انتخاب کرده است.





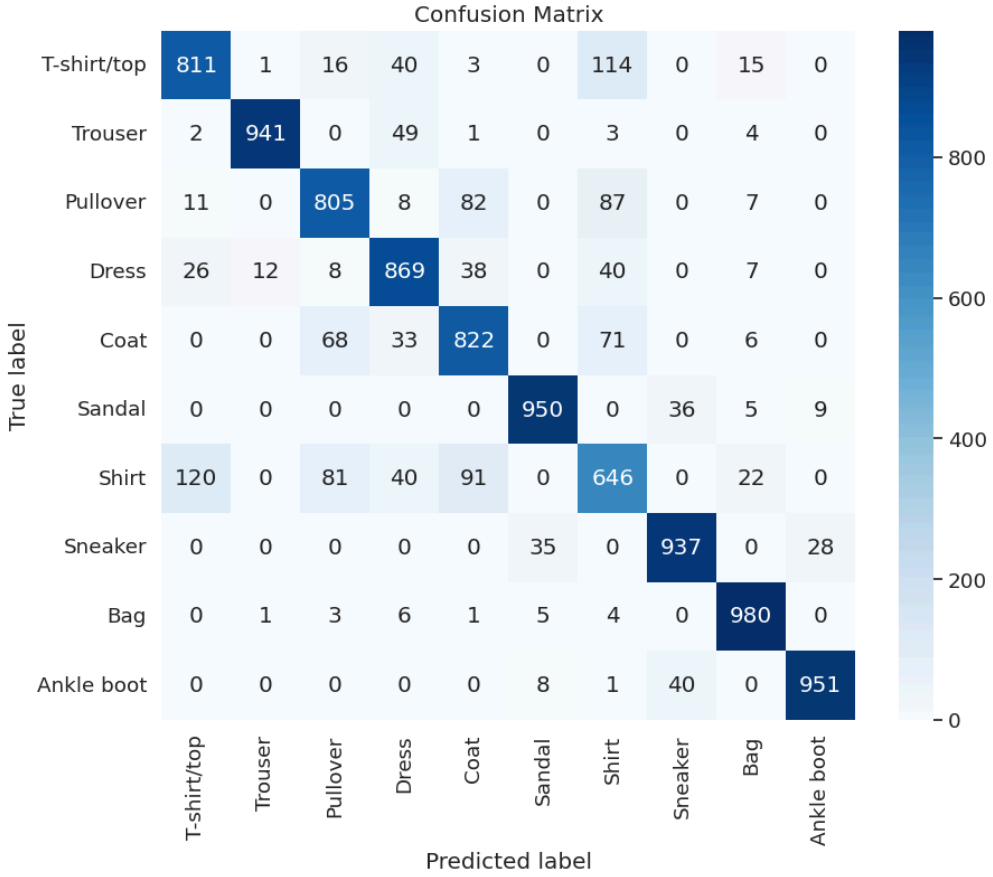
پس ما نیز نتیجه گرفتیم که احتمالا با تقسیم بندی تصاویرمان به ۲ یا ۳ استیت می‌توانیم به جواب خوبی برسیم.

پیاده‌سازی الگوریتم HOG برای رسیدن به دقت بالا کافی نبود. در مقالات مختلف مثل این مقاله دیدیم که معمولا ترکیب چند روش کاهش بعد می‌تواند به ما برای رسیدن به دقت بهتر کمک کند.

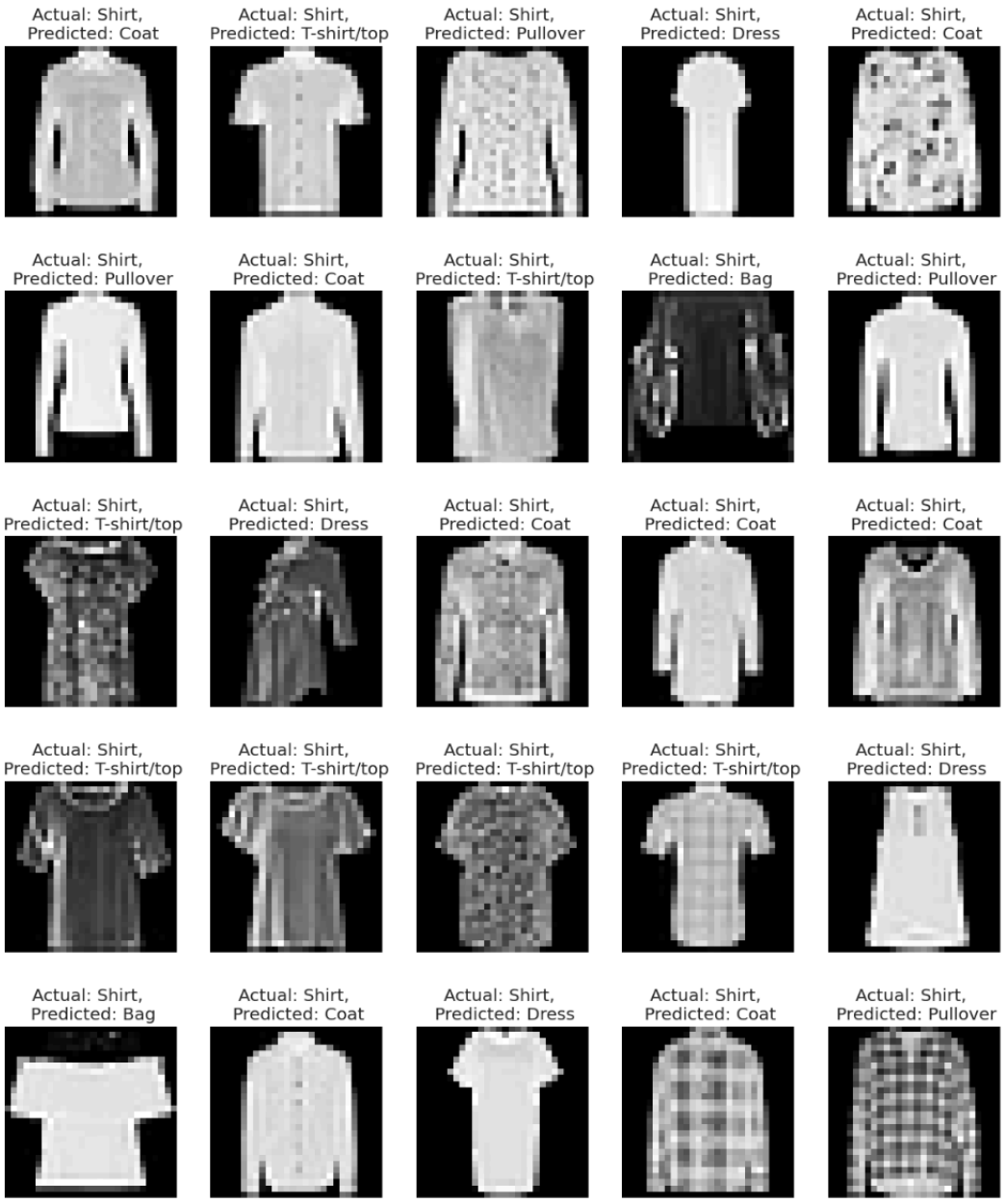
پس ما تصمیم گرفتیم این الگوریتم را با الگوریتم LDA ترکیب کنیم. که دقت حاصله با دو استیت به ۸۶٪ و با یک استیت به ۸۷٪ رسید. با افزایش تعداد استیت های مدل دقت مدل کاهش داشت.

Accuracy: 87.12%

	precision	recall	f1-score	support
T-shirt/top	0.84	0.81	0.82	1000
Trouser	0.99	0.94	0.96	1000
Pullover	0.82	0.81	0.81	1000
Dress	0.83	0.87	0.85	1000
Coat	0.79	0.82	0.81	1000
Sandal	0.95	0.95	0.95	1000
Shirt	0.67	0.65	0.66	1000
Sneaker	0.92	0.94	0.93	1000
Bag	0.94	0.98	0.96	1000
Ankle boot	0.96	0.95	0.96	1000
accuracy			0.87	10000
macro avg	0.87	0.87	0.87	10000
weighted avg	0.87	0.87	0.87	10000

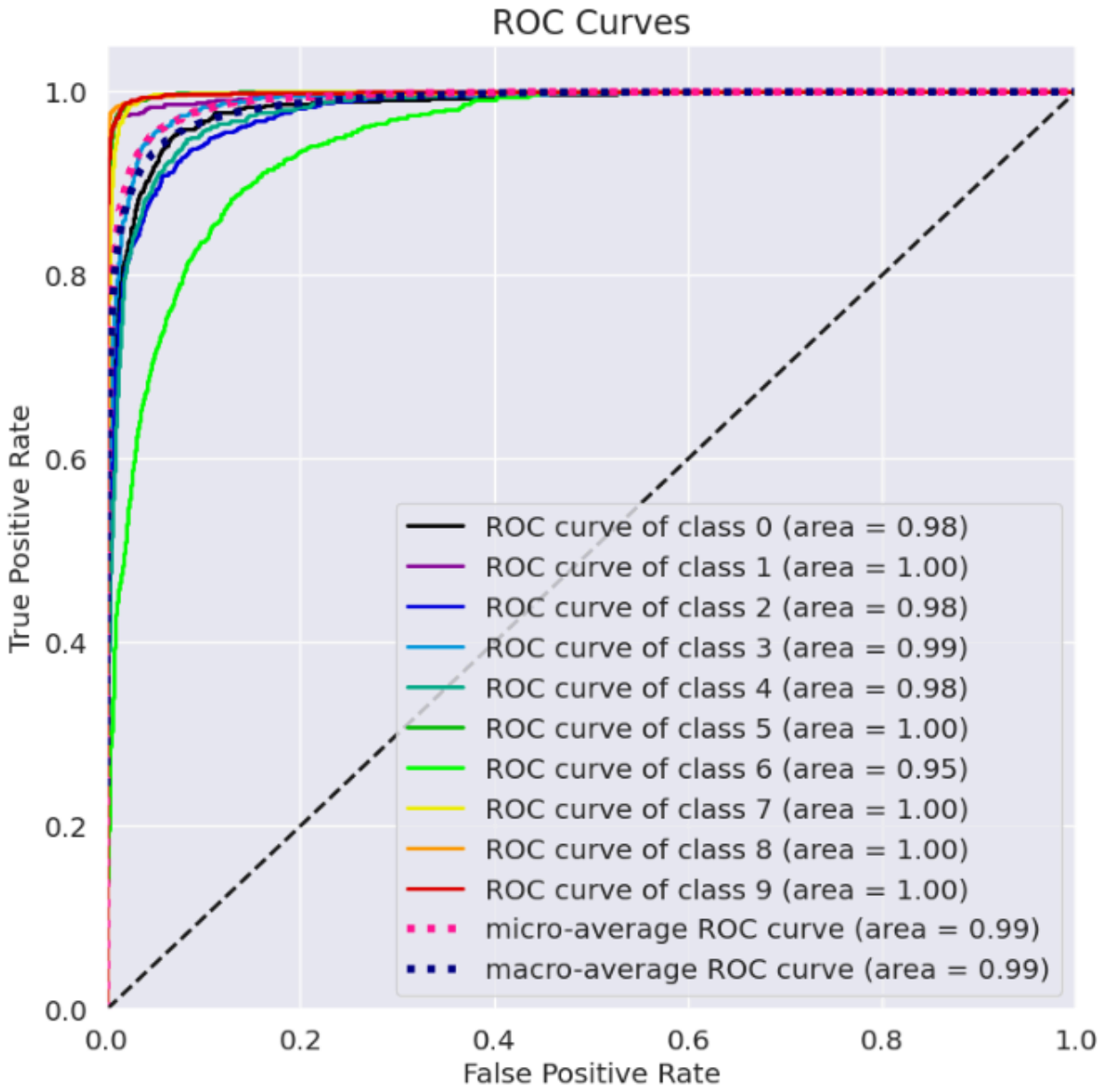


همانطور که میبینیم مدل ما عملکرد خوبی داشته ولی در تشخیص کلاس shirt کمی ضعیف بوده، برای متوجه شدن بهتر دلیل این ضعف پیش‌بینی های این کلاس را نمایش دادیم که به‌صورت زیر است:



البته تشخیص این کلاس در در مدل های پیشرفته‌تر مثل CNN ها نیز به خوبی انجام نمی‌شود زیرا همانطور که میبینیم شامل داده های دشواری است.

درنهایت AUC-ROC این مدل به صورت زیر است:



مسئله ما مولتی کلاس است و برای رسم این نمودار به صورت one vs all استفاده کردیم. هماطور که میبینیم مدل ما توانایی تمایز بخشیدن بسیار خوبی دارد ولی چون این نمودار به صورت one vs all رسم شده، هر کلاس یک threshold بهینه دارد ولی چون ما باید تنها یک threshold کلی در و خوب برای هر ده کلاس درنظر بگیریم، دقت کلی مدل مارا روی ۸۷٪ قرار می‌دهد و نمی‌تواند از این عدد فراتر رود.

نتیجه‌گیری

با اینکه مدل مخفی مارکوف مدلی مرسوم برای انجام تسک طبقه‌بندی تصاویر FashionMNIST نیست، اما با استخراج درست ویژگی ها از تصاویر و معماری درست برای این تسک که می‌تواند train کردن یک مدل به‌ازای هر کلاس باشد، به نتایج قابل قبولی دست پیدا کرد.

در این گزارش مروری بر مسیری که طی شد تا به بیشینه دقت برسیم گفته شده که مهم‌ترین بخش آن استفاده از استخراج‌گر ویژگی HOG و ترکیب آن با LDA بود. همچنین هایپرپارامتر های مسئله از عوامل بسیار مهم در رسیدن به جواب بهینه در این مسئله هستند.