**درس یادگیری ماشین**

# گزارش تکلیف
# Noise Reduction Using PCA and Autoencoders


**استاد درس:**
دکتر افتخاری

**نگارش:**
امیرحسین ابوالحسنی
شماره دانشجویی: ۴۰۰۴۰۵۰۰۳


**پاییز 1403**

# فهرست مطالب

# ۱ مقدمه

یکی از تسک‌هایی که در حوزه پردازش تصویر مطرح می‌شود، کاهش نویز [۱] می‌باشد. در این تسک، سعی بر این می‌شود که با الگوریتم‌ها و راه‌حل‌های متفاوت، نویز تصویر کمتر شود. دو مورد از تکنیک‌های ابتدایی برای این تسک، در این تکلیف بررسی شده و روی دیتاست Fashion MNIST تست می‌شوند که عبارتند از:

- تحلیل مولفه های اصلی [۲]
- خود رمز گذار [۳]

# ۲ درباره دیتاست

مجموعه داده Fashion MNIST یکی از مجموعه‌های پرکاربرد در یادگیری ماشین و بینایی کامپیوتر است. این مجموعه داده شامل تصاویر سیاه و سفید از انواع پوشاک و اکسسوری‌ها می‌باشد که جایگزین مدرنی برای مجموعه داده کلاسیک MNIST است. نمونه ای از این داده‌ها در شکل ۱ قابل مشاهده است.


شکل ۱: بخشی از دیتاست Fashion MNIST

[۱] Noise Reduction
[۲] Principal Component Analysis (PCA)
[۳] Autoencoder

| | |
|---|---|
| تعداد کل تصاویر | ۷۰،۰۰۰ |
| مجموعه آموزش | ۶۰،۰۰۰ |
| مجموعه آزمون | ۱۰،۰۰۰ |
| ابعاد تصاویر | ۲۸×۲۸ پیکسل |
| نوع تصاویر | سیاه و سفید (تک‌کاناله) |
| محدوده مقادیر پیکسل‌ها | ۰ تا ۲۵۵ |

جدول ۱: ویژگی‌های دیتاست Fashion MNIST

این مجموعه داده دارای ۱۰ کلاس مختلف است:

- تی‌شرت/تاپ
- شلوار
- پولیور
- پیراهن
- کت
- صندل
- پیراهن
- کفش ورزشی
- کیف
- چکمه

## ۳ اضافه کردن نویز گاوسی [۱] به دیتاست

نویز گاوسی نوعی نویز می‌باشد که به تصویر اضافه می‌شود. ضابطه زیر، تابع چگالی احتمال این نویز در دو بعد می‌باشد که برای تصاویر به کار برده می شود.

$$n(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma^2}}$$

$\mu$ میانگین
$\sigma$ انحراف معیار که پراکندگی نویز را نشان میدهد
$\sigma^2$ واریانس

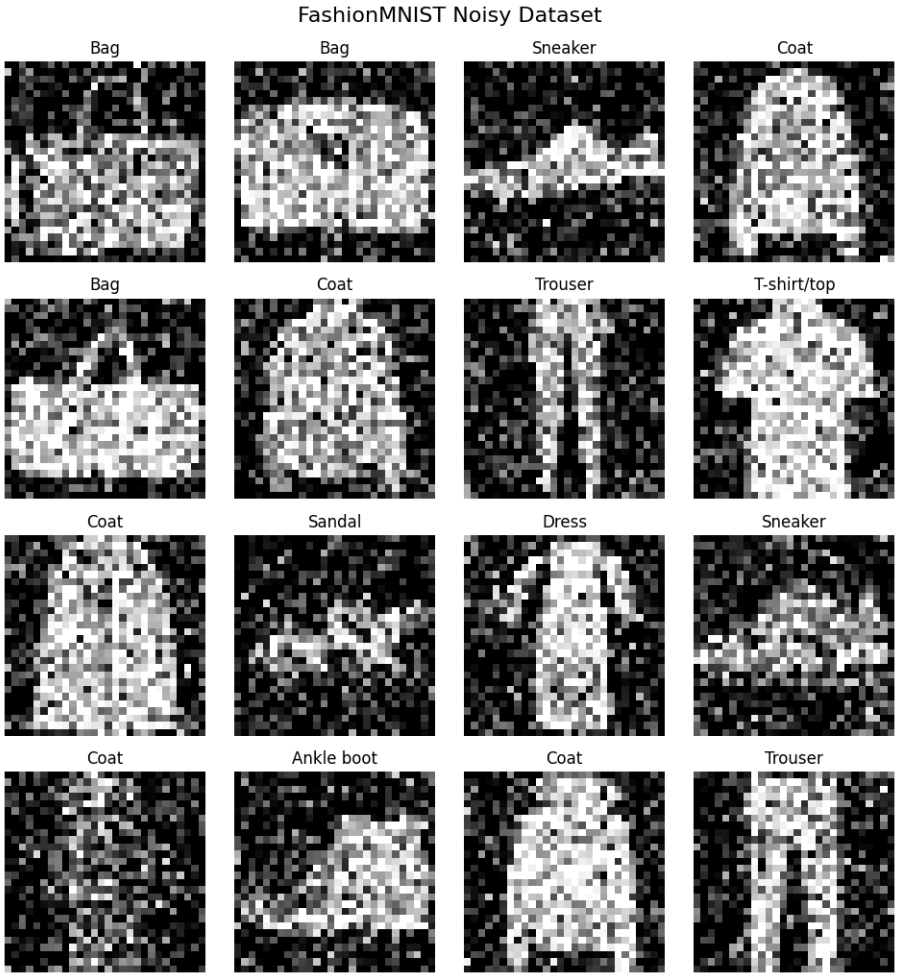
[۱] Gaussian Noise

در پردازش تصویر، فرمول اضافه کردن نویز گاوسی به تصویر به این صورت است:

$$g(x,y) = f(x,y) + n(x,y)$$

$g(x,y)$ تصویر نویزی شده
$f(x,y)$ تصویر اصلی
$n(x,y)$ نویز گوسی که از توزیع نرمال با میانگین و انحراف معیار مشخص پیروی می‌کند

در ابتدا به دیتاست Fashion MNIST نویز گاوسی با واریانس ۰.۱ و میانگین ۰ اضافه می‌کنیم. از این داده‌ها به عنوان داده نویزی در طول تمرین استفاده خواهد شد. [۱] (بخشی از دیتاست در شکل ۲ )




شکل ۲: بخشی از دیتاست نویزی Fashion MNIST

[۱] در اصل هر تصویر به برداری به طول ۷۸۴ تبدیل شده و در این تمرین مورد استفاده قرار می‌گیرد.

# ۴ کاهش نویز با PCA

PCA به عنوان یک روش کاهش بعد خطی، می‌تواند با بدست آوردن مولفه‌های مهم، اطلاعات مهم داده‌ها را حفظ کند. با در نظر گرفتن نویز وارد شده به تصویر به عنوان بخش غیر مهمی از اطلاعات موجود در داده‌ها، انتظار می رود PCA بتواند کاهش نویز روی تصویر انجام دهد. مراحل کار بدین صورت است که ابتدا ماتریس $Q$ [۱] که شامل بردارهای eigen می باشد را برای ماتریس کواریانس $\Sigma$ بدست می آوریم:

$$\Sigma = Q\Lambda Q^{-1}$$

$$Q = [q_1\ q_2\ \cdots\ q_d]$$

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$$

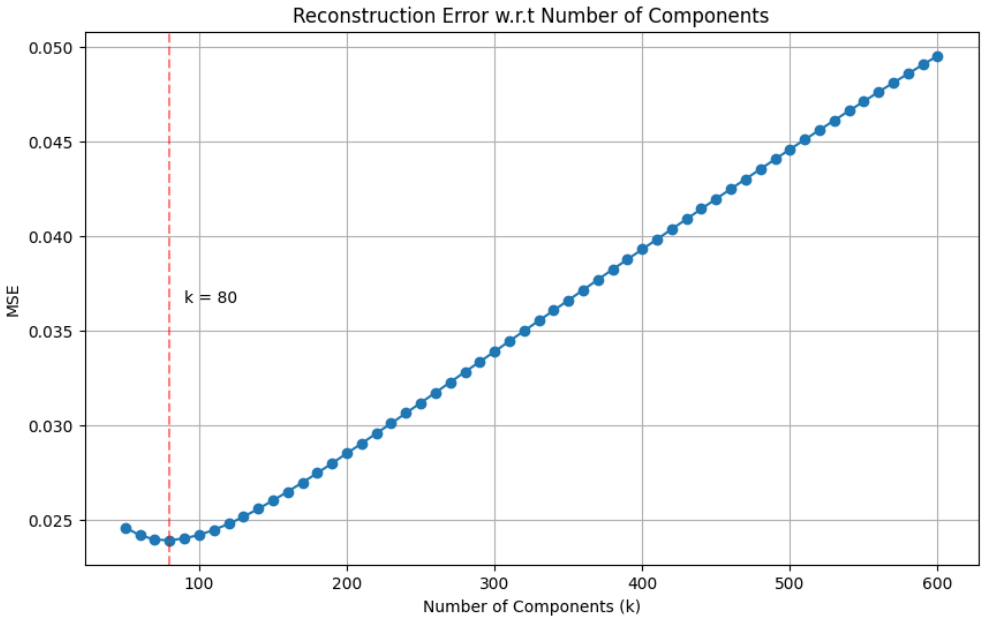
سپس داده‌ها را بر روی مولفه‌های اصلی تصویر می‌کنیم:

$$Y = (X - \mu)Q$$

در نهایت داده‌ها را به بعد اصلی‌شان بر می‌گردانیم:

$$X' = YQ^T + \mu$$

انتظار می رود با انتخاب مقدار مناسبی به عنوان تعداد مولفه‌های اصلی در نظر گرفته شده، این کاهش بعد و سپس بازگردانی داده‌ها به تعداد بعد اولیه باعث حذف مقدار زیادی نویز از تصویر بشود.
در تکلیف ارائه شده، مقدار ۸۰ مولفه اصلی بهترین مقدار از بین ۷۸۴ مقدار ممکن برای تعداد مولفه‌های اصلی بوده است. این مقدار به ما خطای ۰.۰۲۳۹ را می‌دهد که بهترین خطای بازسازی در بین دیگر مقادیر مولفه‌های اصلی می باشد.(شکل ۳ )




شکل ۳: پیدا کردن بهترین مقدار کاهش بعد با توجه به خطای بازسازی

[۱] ماتریس $Q$ شامل بردار های یکه و متعامد به یکدیگر می‌باشد. و بدین علت $Q^{-1} = Q^T$

# ۵ کاهش نویز با Autoencoder

یک Autoencoder، یک نوع شبکه عصبی می‌باشد که از دو بخش Encoder و Decoder تشکیل شده است. این نوع شبکه‌های عصبی می‌توانند به نوعی کاهش بعد خطی (در صورت داشتن تابع فعال ساز خطی) و یا غیر خطی (در صورت داشتن تابع فعال ساز غیر خطی) انجام دهند.

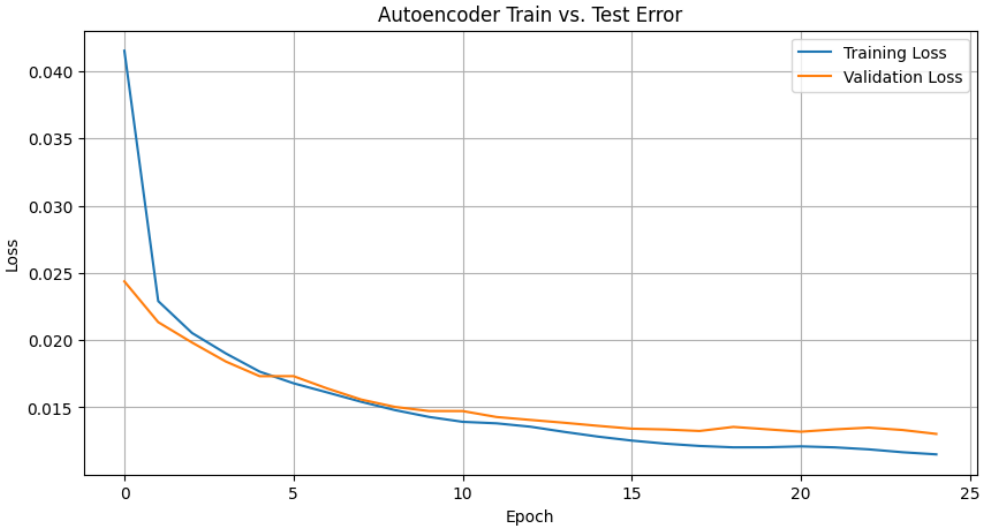
یکی از استفاده‌هایی که از این نوع معماری شبکه می توان انجام داده این است که در ورودی به آن تصویر نویزی داده شود و برچسب هر تصویر نویزی، تصویر بدون نویز باشد. بدین ترتیب Autoencoder آموزش می‌بیند تا نگاشتی از تصویر نویزی به تصویر بدون نویز برقرار کند. که این همان مدلی می‌باشد که ما به آن نیاز داریم.

شبکه‌ای که با فریم ورک پایتورچ متشکل از چند لایه Dense و تابع فعال ساز ReLU پیاده سازی شد. تعداد نورون‌های هر لایه به صورت زیر است:

$$\text{Encoder} : 784 \rightarrow 512 \rightarrow 256 \rightarrow 128$$

$$\text{Decoder} : 128 \rightarrow 256 \rightarrow 512 \rightarrow 784$$

در شکل ۴ رفتار خطای آموزش و تست دیده می‌شود که دال بر یادگیری مناسب بدون بیش برازش [1] می‌باشد. با این روش و هایپر پارامترهای گفته شده در جدول ۲ به خطای ۰.۰۱۳ رسیده شد.




شکل ۴: رویه خطای آموزش و تست Autoencoder

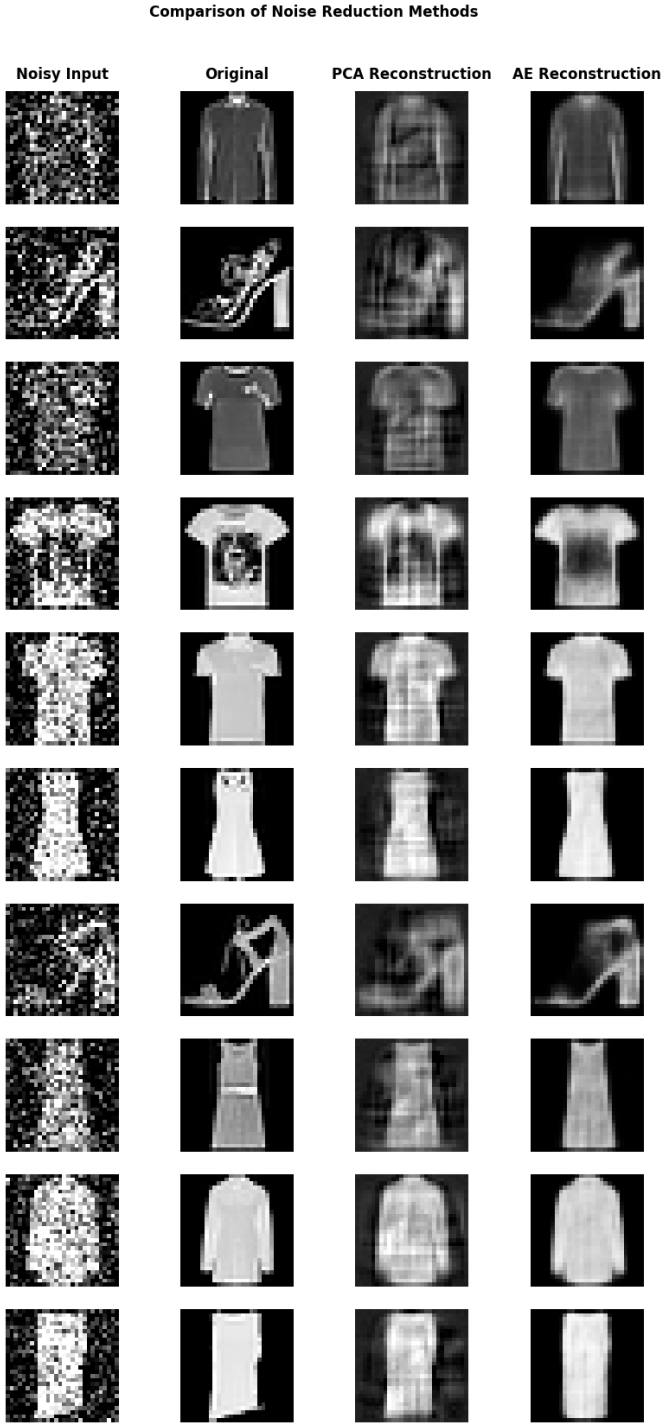
| | |
|---|---|
| نرخ یادگیری | ۰.۰۰۱ |
| سایز هر دسته | ۱۲۸ |
| Epochs | ۲۵ |

جدول ۲: هایپر پارامترهای Autoencoder

[1] Overfit

# ۶ نتایج

با مقایسه خطای گزارش شده در بخش‌های ۵ و ۴ انتظار می‌رود کاهش نویز با Autoencoder به نسبت PCA بیشتر باشد. در شکل ۵ می‌توان مقایسه ای بین تصویر نویزی، تصویر اصلی، کاهش نویز توسط PCA و کاهش نویز توسط Autoencoder را مشاهده کرد.




شکل ۵: رویه خطای آموزش و تست Autoencoder

## ۷ چالش: مقایسه عادلانه!

بزرگترین چالشی که در پیاده سازی تکلیف به آن برخورده شد، این بود که دیتاست بایستی از یک منبع دانلود شده و سپس با یکبار اضافه کردن نویز، در کل تمرین و توسط هر روش استفاده و خطای روی آن گزارش می‌شد. این بدین علت بود که کتابخانه *Scikit Learn* برای متد PCA و برای آموزش Autoencoder از کتابخانه *PyTorch* استفاده شده است. و اگر هر بار نویز گاوسی به هر یک از دیتاست ها برای هر متد اضافه می‌شود، تضمینی بر یکسانی نویز اضافه شده به تصویر نبود و همچنین لود کردن دیتاست از روی هر کتابخانه برای هر متد منجر به افزونگی نیز می‌گردید. برای همین، تنها یکبار دیتاست دانلود شد و در همان ابتدا نویز به آن اضافه شد و در کل تمرین مورد استفاده قرار گرفت تا در انتها مقایسه عادلانه تضمین گردد.

## ۸ نتیجه‌گیری

با توجه به نتایج بدست آمده در بخش ۶ می‌توان به این نتیجه رسید که هر دو متد برای کاهش نویز کارساز بوده اند، اما متد Autoencoder بهتر از PCA توانسته کاهش نویز را انجام دهد. که بنظر می‌رسد غیر خطی بودن کاهش بعد Autoencoder این قدرت را به مدل می‌دهد که روابط غیرخطی مهم را نیز در نظر بگیرد و تنها به روابط خطی (به مانند PCA) بسنده نکند.