دانشکده مهندسی کامپیوتر

گزارش پایانی درس: مبانی هوش محاسباتی

عنوان پژوهش: تمرین سوم بخش دوم (شبکه‌ی CNN)

ارائه دهنده:

نرگس سادات موسوی جد

شادی شاهی محمدی

استاد درس:

دکتر کارشناس

بهار ۱۴۰۴

فهرست:

۱–تشریح کد:

۱–۱–ساختار پیشنهاد شده‌ی CNN:

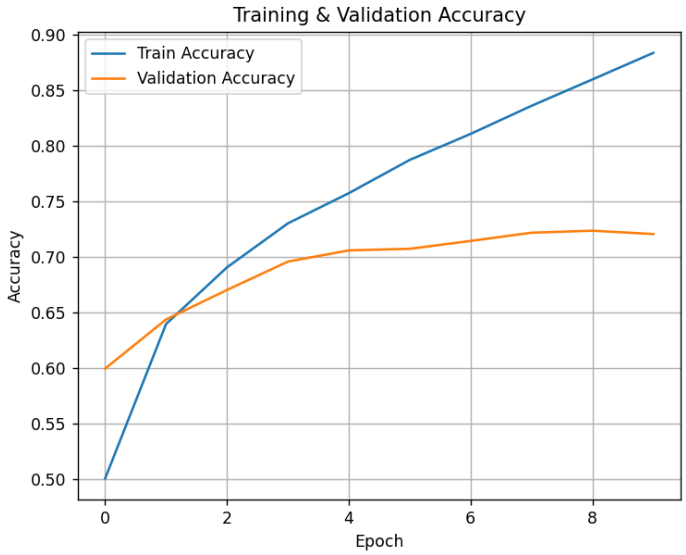
مدل cnnHyper یک معماری ساده و ابتدایی شبکه عصبی کانولوشنی (CNN) است که برای دسته‌بندی تصاویر مجموعه داده CIFAR-10 طراحی شده است. این مدل شامل دو لایه‌ی کانولوشن است که به ترتیب ۳۲ و ۶۴ فیلتر ۳×۳ دارند و از تابع فعال‌سازی ReLU و پدینگ 'same' استفاده می‌کنند تا ابعاد فضایی ویژگی‌ها حفظ شود. پس از هر لایه‌ی کانولوشن، یک لایه‌ی MaxPooling با اندازه ۲×۲ قرار دارد که برای کاهش ابعاد و جلوگیری از بیش‌برازش به کار می‌رود.

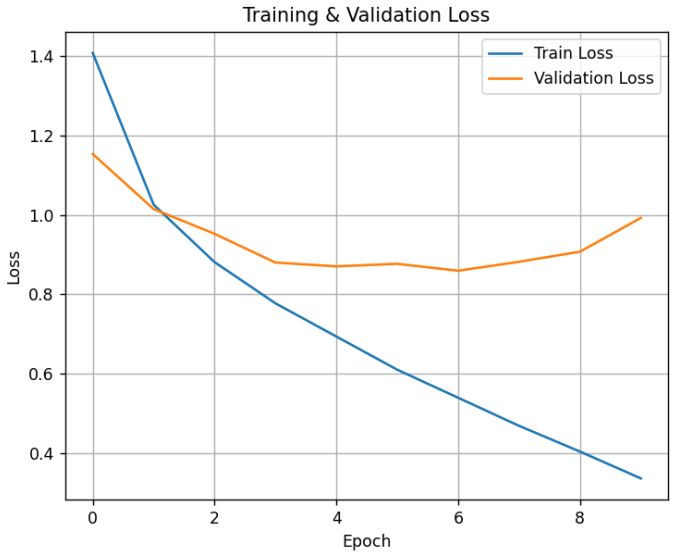
پس از استخراج ویژگی‌ها از تصویر، داده‌ها توسط لایه‌ی Flattenبه یک بردار یک‌بعدی تبدیل می‌شوند تا آماده‌ی ورود به لایه‌های پرسپترون (Dense) شوند. یک لایه Dense با ۱۲۸ نورون و تابع ReLU به‌عنوان لایه پنهان استفاده شده و در نهایت لایه خروجی با ۱۰ نورون و تابع softmax قرار دارد که احتمال تعلق تصویر به هرکدام از ۱۰ کلاس داده را پیش‌بینی می‌کند.

در آخر مدل با بهینه‌ساز Adam و تابع زیان sparse_categorical_crossentropyکامپایل شده است. معیار ارزیابی نیز دقت (accuracy) است. آموزش مدل با ۱۰ دوره‌ی تکرار (epoch) و اندازه دسته‌ای ۶۴ انجام شده است.

```
def cnn_struct1(x_train, y_train, x_test, y_test):
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', padding='same',
input_shape=(32, 32, 3)),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dense(10, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    history = model.fit(x_train, y_train, epochs=10,
                        validation_data=(x_test, y_test),
                        batch_size=64)

    return model, history
```

نمودار دقت و خطا و سایر نتایج این شبکه به صورت زیر است:






```
Classification Report:
              precision    recall  f1-score   support

           0     0.6833    0.8200    0.7455      1000
           1     0.8112    0.8290    0.8200      1000
           2     0.7538    0.4990    0.6005      1000
           3     0.5813    0.4720    0.5210      1000
           4     0.6501    0.7320    0.6886      1000
           5     0.5991    0.6470    0.6221      1000
           6     0.7361    0.7920    0.7630      1000
           7     0.7679    0.7810    0.7744      1000
           8     0.8383    0.8190    0.8285      1000
           9     0.7928    0.8150    0.8037      1000

    accuracy                         0.7206     10000
   macro avg     0.7214    0.7206    0.7167     10000
weighted avg     0.7214    0.7206    0.7167     10000

Macro-average F1 Score: 0.7167318758595042
```

۱–۲–CNN با یک لایه اضافه‌تر:

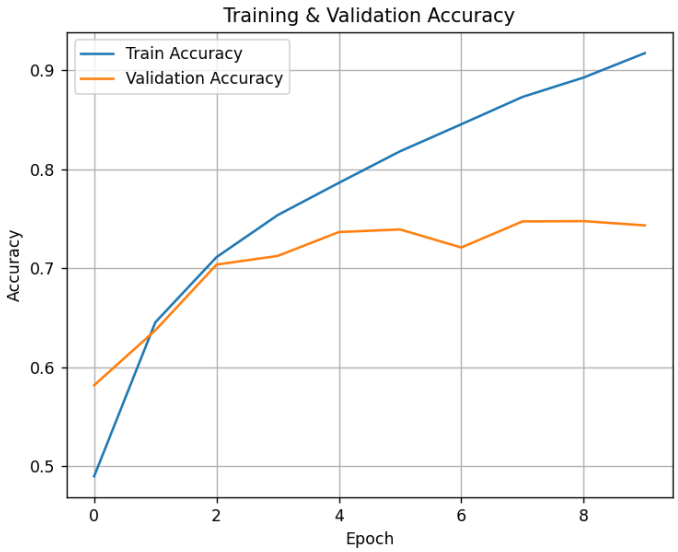
مدل cnn_struct2 نسبت به مدل قبلی، تغییرات مهمی در افزایش عمق شبکه و تعداد پارامترهای قابل یادگیری کرده است. این تغییرات به‌طور خاص در اضافه شدن یک لایه‌ی کانولوشنی جدید با ۱۲۸ فیلتر ۳×۳ نمایان است که بعد از دو لایه‌ی کانولوشنی اولیه قرار گرفته است. همچنین، اندازه لایه‌ی Dense نیز از ۱۲۸ به ۲۵۶ نورون افزایش یافته که توان مدل برای یادگیری الگوهای پیچیده‌تر را بیشتر می‌کند.

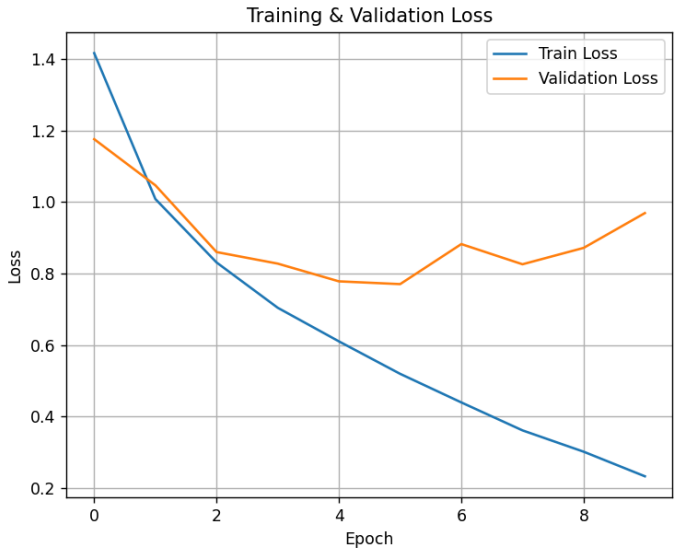
لایه‌ی سوم کانولوشن (با ۱۲۸ فیلتر) باعث می‌شود که ویژگی‌های عمیق‌تری از تصویر استخراج شوند، مخصوصاً ویژگی‌هایی که در سطح بالا (high-level) قرار دارند. این موضوع برای داده‌هایی مثل CIFAR-10 که شامل تصاویر پیچیده از کلاس‌های مختلف است، می‌تواند باعث افزایش توان مدل در تشخیص تفاوت‌ها شود.

نتیجه‌ی این تغییرات معمولاً به صورت افزایش دقت روی داده‌های آموزشی و تست مشاهده می‌شود، به‌ویژه اگر مدل به‌درستی تنظیم شده باشد. با این حال، افزایش عمق شبکه همچنین خطر **بیش‌برازش** (overfitting) را بالا می‌برد.

```
def cnn_struct2(x_train, y_train, x_test, y_test):
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', padding='same',
input_shape=(32, 32, 3)),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dense(10, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    history = model.fit(x_train, y_train, epochs=10,
                        validation_data=(x_test, y_test),
                        batch_size=64)
    return model, history
```

نمودار دقت و خطا و سایر نتایج برای این شبکه به صورت زیر است. همان طور که مشاهده می‌شود نتایج با اضافه کردن این لایه بهبود یافته‌اند و به نظر نمی‌رسد بیش‌برازش رخ داده باشد.






```
Classification Report:
              precision    recall  f1-score   support

           0     0.7956    0.7940    0.7948      1000
           1     0.8730    0.8110    0.8409      1000
           2     0.6746    0.6840    0.6792      1000
           3     0.5299    0.5850    0.5561      1000
           4     0.7518    0.6240    0.6820      1000
           5     0.6352    0.6460    0.6406      1000
           6     0.8418    0.7770    0.8081      1000
           7     0.7365    0.8300    0.7804      1000
           8     0.8505    0.8360    0.8432      1000
           9     0.7851    0.8440    0.8135      1000

    accuracy                         0.7431     10000
   macro avg     0.7474    0.7431    0.7439     10000
weighted avg     0.7474    0.7431    0.7439     10000

Macro-average F1 Score: 0.7438711558885629
```

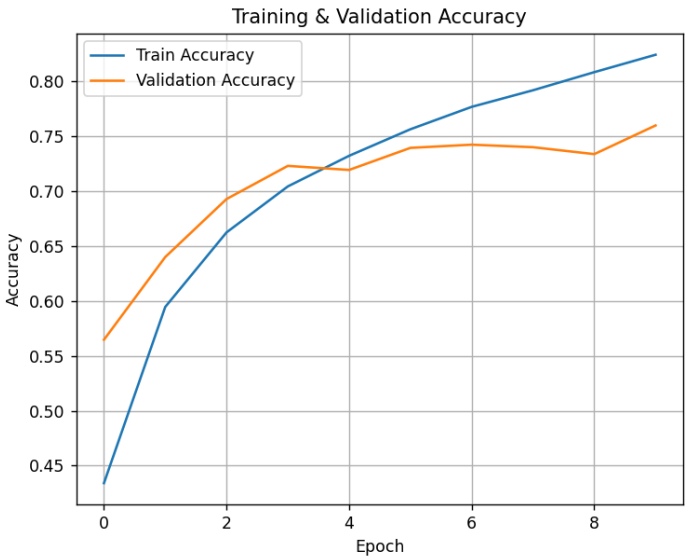
۱-۳-CNN با تکنیک منظم سازی Dropout:

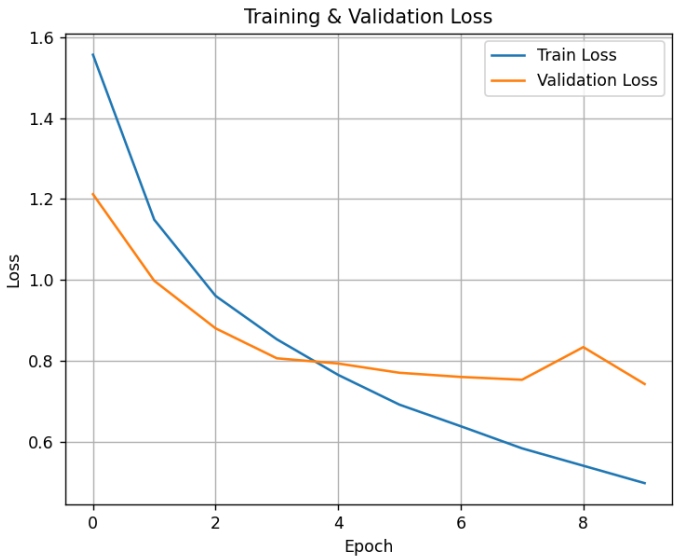
در مدل cnn_struct3 ساختار کلی مشابه مدل cnn_struct2 است، اما یک تغییر مهم در آن اعمال شده است که استفاده از لایه‌ی Dropout برای کاهش بیش‌برازش می‌باشد. این روش یکی از تکنیک‌های مؤثر در منظم‌سازی (Regularization) شبکه‌های عصبی است که در طول آموزش، به‌طور تصادفی برخی نورون‌ها را غیرفعال می‌کند تا شبکه وابستگی شدید به نورون‌های خاص پیدا نکند.

در این مدل، لایه‌ی Dropout(0.5) بعد از لایه‌ی Dense(256) قرار گرفته است. این به این معناست که در هر بار آموزش، به‌طور تصادفی ۵۰٪ از نورون‌های لایه‌ی Dense غیرفعال می‌شوند. هدف از این کار، افزایش تعمیم‌پذیری مدل (Generalization) است، یعنی اینکه مدل روی داده‌های جدید (که در آموزش دیده نشده‌اند) عملکرد بهتری داشته باشد و فقط الگوهای خاص داده‌های آموزشی را یاد نگیرد.

```
def cnn_struct3(x_train, y_train, x_test, y_test):
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', padding='same',
input_shape=(32, 32, 3)),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(10, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    history = model.fit(x_train, y_train, epochs=10,
                        validation_data=(x_test, y_test),
                        batch_size=64)
    return model, history
```

نتایج حاص از این روش نیز نشان‌دهنده‌ی افزایش دقت مدل روی داده‌های تست می‌باشد. نمودار و نتایج این روش بدین شرح است:






```
Classification Report:
              precision    recall  f1-score   support

           0     0.7797    0.8210    0.7998      1000
           1     0.8988    0.8350    0.8657      1000
           2     0.6424    0.6790    0.6602      1000
           3     0.5429    0.6460    0.5900      1000
           4     0.7156    0.7170    0.7163      1000
           5     0.7193    0.5970    0.6525      1000
           6     0.8198    0.8190    0.8194      1000
           7     0.8053    0.8270    0.8160      1000
           8     0.8619    0.8550    0.8584      1000
           9     0.8697    0.8010    0.8339      1000

    accuracy                         0.7597     10000
   macro avg     0.7655    0.7597    0.7612     10000
weighted avg     0.7655    0.7597    0.7612     10000

Macro-average F1 Score: 0.7612188818321657
```

۱-۴- CNN با تکنیک منظم سازی Data Augmentation:

در این مدل، تغییر اصلی نسبت به نسخه‌ی قبلی، استفاده از تکنیک افزایش داده است. این تکنیک قبل از ورود تصویر به لایه‌های کانولوشن اجرا می‌شود و هدف آن افزایش تنوع تصاویر آموزشی بدون نیاز به جمع‌آوری داده‌های بیشتر است. این تغییر نقش بسیار مهمی در بهبود تعمیم‌پذیری مدل دارد.

در این مدل، یک لایه‌ی Sequential برای افزایش داده تعریف شده که شامل چهار عملگر است:

۱. **RandomFlip("horizontal")**: به‌طور تصادفی تصویر را در محور افقی برمی‌گرداند.

۲. **RandomRotation(0.1)**: چرخش تصادفی تصویر به میزان حداکثر ۱۰٪.

۳. **RandomZoom(0.1)**: زوم تصادفی به اندازه ۱۰٪ در تصویر.

۴. **RandomContrast(0.1)**: تغییرات تصادفی در کنتراست تصویر تا ۱۰٪.

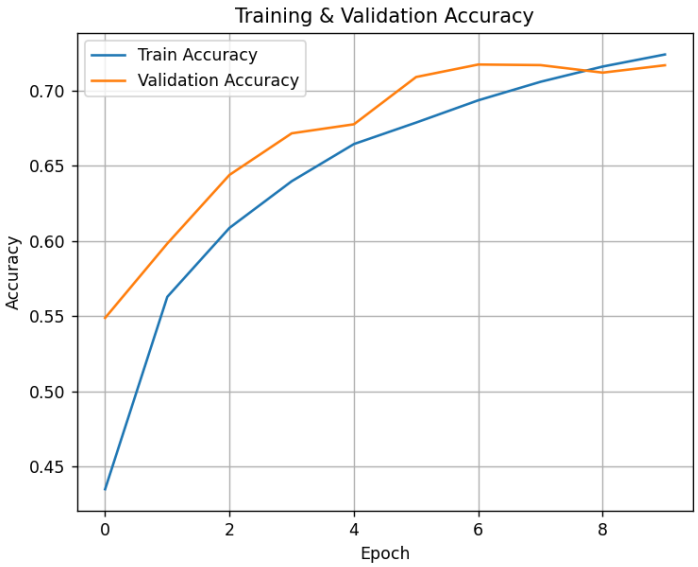
این تغییرات به مدل کمک می‌کنند تا نسبت به نویز و تغییرات ظاهری در تصاویر مقاوم‌تر شده و تنها ویژگی‌های مهم و پایدار را یاد بگیرد.

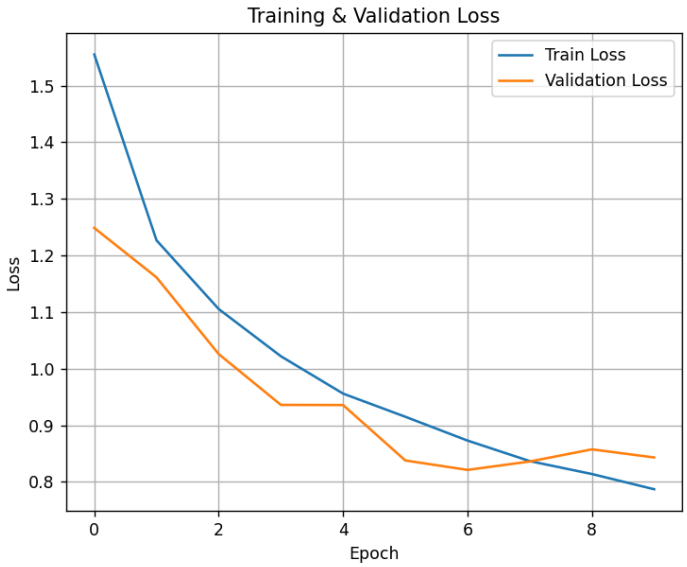
تأثیر استفاده از Data Augmentation معمولاً به صورت بهبود دقت در داده‌های تست و کاهش احتمال بیش‌برازش دیده می‌شود. اگرچه ممکن است دقت اولیه‌ی مدل کندتر افزایش یابد، اما در پایان، مدل عملکرد پایدارتر و قابل‌اعتمادتری روی داده‌های نادیده‌گرفته شده (مثل داده‌های تست) از خود نشان می‌دهد.

```
def cnn_struct4(x_train, y_train, x_test, y_test):
    # (Data Augmentation)
    data_augmentation = tf.keras.Sequential([
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.1),
        layers.RandomContrast(0.1),
    ])

    model = models.Sequential([
        data_augmentation,

        layers.Conv2D(32, (3, 3), activation='relu', padding='same',
input_shape=(32, 32, 3)),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
```

```
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),

        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dense(10, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    history = model.fit(x_train, y_train,
                        epochs=10,
                        validation_data=(x_test, y_test),
                        batch_size=64)

    return model, history
```

نتایج و نمودارهای این شبکه نیز به صورت زیر می‌باشد. همانطور که مشاهده می‌شود نتایج نسبت به حالت قبلی بدتر شده بنابراین بهتری شبکه در بین موارد ذکر شده حالت سوم می‌باشد.





```
Classification Report:
              precision    recall  f1-score   support

           0     0.7774    0.7650    0.7712      1000
           1     0.7694    0.8910    0.8258      1000
           2     0.7168    0.5620    0.6300      1000
           3     0.5687    0.5670    0.5679      1000
           4     0.7456    0.5570    0.6377      1000
           5     0.7609    0.4870    0.5939      1000
           6     0.6438    0.8620    0.7371      1000
           7     0.7375    0.8090    0.7716      1000
           8     0.8280    0.8090    0.8184      1000
           9     0.6719    0.8580    0.7536      1000

    accuracy                         0.7167     10000
   macro avg     0.7220    0.7167    0.7107     10000
weighted avg     0.7220    0.7167    0.7107     10000

Macro-average F1 Score: 0.7107078070716369
```

۱–۵–مزایا و معایب CNN در این مثال نسبت به پرسپترون:

در مسئله‌ی طبقه‌بندی تصاویر مانند مجموعه داده‌ی CIFAR-10، استفاده از شبکه‌های عصبی کانولوشنی (CNN) در مقایسه با پرسپترون چندلایه (MLP) دارای مزایا و معایب خاص خود است که در ادامه به آن‌ها می‌پردازیم:

مزایای CNN نسبت به MLP:

درک بهتر از ساختار فضایی تصویر: CNNها به‌طور خاص برای داده‌های تصویری طراحی شده‌اند و به کمک لایه‌های کانولوشن می‌توانند ویژگی‌های محلی مثل لبه‌ها، بافت‌ها و الگوهای فضایی را شناسایی کنند. در حالی که MLP تصاویر را به یک بردار یک‌بعدی تبدیل می‌کند و اطلاعات مکانی بین پیکسل‌ها را از بین می‌برد.

تعداد پارامتر کمتر و کارایی بیشتر: لایه‌های کانولوشنی از پارامترهای مشترک (shared weights) استفاده می‌کنند، در حالی که MLP برای هر نورون در هر لایه پارامتر جداگانه‌ای دارد. این موضوع باعث می‌شود CNNها حتی با عمق بالا، نسبت به MLPها بسیار کم‌حجم‌تر و بهینه‌تر باشند.

مقیاس‌پذیری بهتر: CNNها می‌توانند به‌راحتی روی تصاویر با اندازه‌های بزرگ‌تر یا مجموعه‌داده‌های پیچیده‌تر گسترش یابند، در حالی که MLPها با افزایش ابعاد داده به‌شدت دچار انفجار پارامتر می‌شوند.

تعمیم‌پذیری قوی‌تر: به دلیل ساختار سلسله‌مراتبی یادگیری ویژگی‌ها، شبکه‌های CNN توانایی بالاتری در یادگیری ویژگی‌های معنادار و تعمیم دادن آن‌ها به داده‌های جدید دارند.

معایب CNN نسبت به MLP:

پیچیدگی بیشتر در طراحی: معماری CNN شامل انتخاب‌های متنوعی مانند اندازه فیلتر، تعداد فیلتر، نوع pooling، stride، padding و غیره است که نیاز به تنظیم دقیق (tuning) دارند. در مقابل، ساختار MLP ساده‌تر و مستقیم‌تر است.

نیاز به سخت‌افزار قوی‌تر: به دلیل عملیات‌های پیچیده کانولوشن و عمق بیشتر مدل‌ها، CNNها نیاز به منابع محاسباتی بیشتر (مثل GPU) دارند، در حالی که MLPها با منابع ساده‌تری نیز قابل اجرا هستند.

وابستگی به داده‌های ساختاریافته: CNNها برای داده‌هایی که ساختار دوبعدی یا مکانی دارند (مثل تصاویر) عالی هستند، اما برای داده‌های بدون ساختار فضایی (مثل داده‌های جدولی یا متنی) مناسب نیستند. در آن موارد، MLP یا سایر مدل‌ها انتخاب بهتری هستند.

لینک گیت هاب: https://github.com/Rshshad/neural_network_homework3-