مدرس: **دکتر فدایی**
طراح: حسام اسداله‌زاده، ادیب رضایی، مسعود طهماسبی‌فرد، نوید دهبان

مهلت تحویل: جمعه ۱۹ خرداد ۱۴۰۲، ساعت ۲۳:۵۹

مقدمه

شبکه‌های عصبی کانولوشنی - Convolutional Neural Networks (CNNs)

شبکه‌های عصبی کانولوشنی یا پیچشی (CNN) نوعی مدل یادگیری عمیق هستند که به طور گسترده و به خصوص برای پردازش تصویر و ویدیو استفاده می‌شوند. این مدل‌ها برای تجزیه و تحلیل و شناسایی موثر الگوهای بصری با استفاده از مفهوم Convolution طراحی شده‌اند. CNN ها معمولا از چندین لایه شامل لایه‌های کانولوشن، لایه‌های ادغام[1] و لایه‌های کاملا متصل[2] تشکیل می‌شوند. لایه‌های کانولوشن، فیلترهایی را روی تصویر ورودی اعمال می‌کنند و ویژگی‌ها را در مکان‌های مختلف تصویر یا ویدیو استخراج می‌کنند. لایه‌های ادغام ابعاد فضایی ویژگی‌ها را کاهش می‌دهند و لایه‌های کاملاً متصل بر اساس ویژگی‌های استخراج شده، عملیات طبقه‌بندی را انجام می‌دهند. این معماری سلسله مراتبی، CNN ها را قادر می‌سازد تا الگوهای بصری پیچیده را به طور خودکار یاد بگیرند. این قابلیت مدل‌های کانولوشنی، آنها را در کارهایی مانند طبقه‌بندی تصویر، تشخیص اشیا و بخش‌بندی[3] تصویر بسیار پرقدرت می‌سازد.

CNN ها در حوزه‌های مختلف به موفقیت‌های بزرگی دست یافته‌اند و به پیشرفت در زمینه‌ی بینایی ماشین کمک چشمگیری کرده‌اند. تاریخچه این شبکه‌ها به سال‌های ۱۹۸۰ و ۱۹۹۰ برمی‌گردد، زمانی که با وجود فراگیر شدن رایانه‌ها و روش‌های جدید پردازش تصویر مانند تبدیل فوریه و Small Discrete Fourier Transform و ... همچنان محدودیت‌های زیادی در این حوزه وجود داشت. اما با پیدایش شبکه‌های کانولوشنی، این مدل‌ها توانستند با نوآوری‌های خود در زمینه تشخیص الگو و تصویر، به پردازش تصویر به صورت خودکار و با کارایی بالا بپردازند.

یکی از اولین مدل‌های شبکه کانولوشنی، مدل LeNet نام دارد که توسط یان لی‌کان (Yann LeCun) در سال ۱۹۹۸ ارائه شد. این مدل برای تشخیص و تمایز اعداد دست‌نوشته مورد استفاده قرار گرفت و در مسابقات مشهور تشخیص اعداد دست‌نوشته (MNIST) با استفاده از کامپیوتر به خوبی عمل کرد[4]. مدل LeNet از لایه‌های کانولوشنی، لایه‌های ادغام و لایه‌های کاملا متصل تشکیل شده‌اند. این مدل با استفاده از ترکیب این لایه‌ها و استخراج ویژگی‌های هر یک از اعداد، قادر به تشخیص و تمایز آن‌ها می‌شود. LeNet بنیان‌گذار و الگوی اصلی برای توسعه شبکه‌های کانولوشنی در سال‌های بعدی بود و نقش مهمی در پیشرفت علوم و تکنولوژی مرتبط با بینایی ماشین و تشخیص الگو ایفا کرد.

یکی از رویدادهای تاریخی قابل توجه در توسعه CNN ها، چالش تشخیص تصویری در مقیاس بزرگ ImageNet[5] در سال 2012 بود. مدل برنده، معروف به AlexNet، یک CNN عمیق بود که به طور قابل توجهی از الگوریتم‌های بینایی ماشین سنتی بهتر عمل کرد.

---

[1] Pooling

[2] Fully-Connected

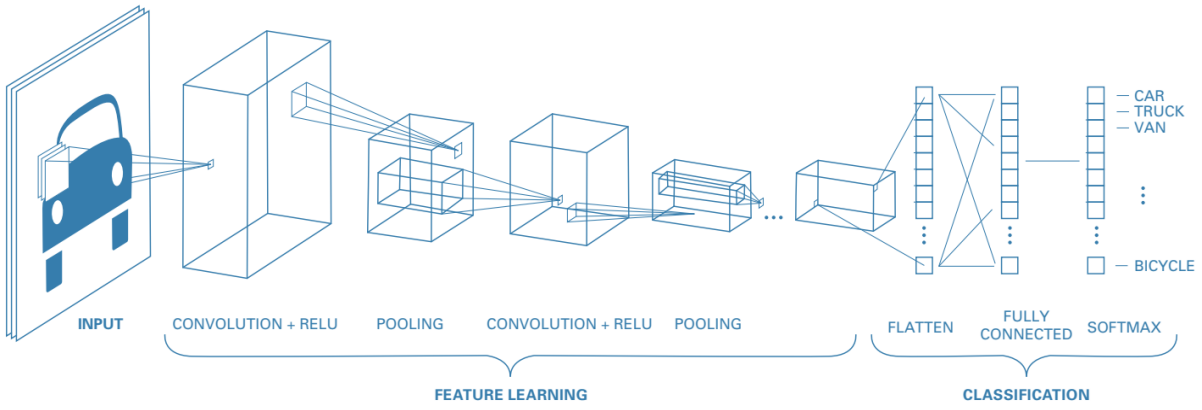
[3] Segmentation

[4] Convolutional Network Demo from 1989

[5] ImageNet Large Scale Visual Recognition Challenge

موفقیت AlexNet، پتانسیل CNN ها را در وظایف طبقه‌بندی تصاویر نشان داد و باعث ایجاد علاقه بیشتر و در نتیجه گسترش پژوهش‌ها در این زمینه شد.

از آن زمان، CNN ها به پیشرفت‌های متعددی دست یافته‌اند. در سال 2015، مدل VGGNet عملکرد چشمگیری را در وظایف طبقه‌بندی تصاویر نشان داد و معماری آن با لایه‌های کانولوشنی متعدد به طور گسترده‌ای مورد استفاده قرار گرفت. نقطه عطف دیگر، معرفی معماری ResNet در سال 2015 بود که به چالش آموزش شبکه‌های بسیار عمیق پرداخت و منجر به پیشرفت‌های قابل توجهی در دستیابی به مدل‌هایی با دقت بالا شد. علاوه بر این، CNN ها در پیشرفت تکنیک‌های تشخیص اشیا، با مدل‌هایی مانند Faster R-CNN که به سرعت و دقت قابل‌توجهی دست می‌یابند، نقش مهمی داشته‌اند. CNN ها با تمرکز بر حوزه‌هایی مانند مکانیسم‌های توجه[6]، یادگیری انتقالی[7] و تفسیرپذیری[8] به تکامل خود ادامه می‌دهند تا مرزهای وظایف بینایی ماشین را حتی بیش از این، پیش ببرند.



شکل ۱. ساختار کلی یک شبکه‌ی کانولوشنی

[6] Attention

[7] Transfer Learning

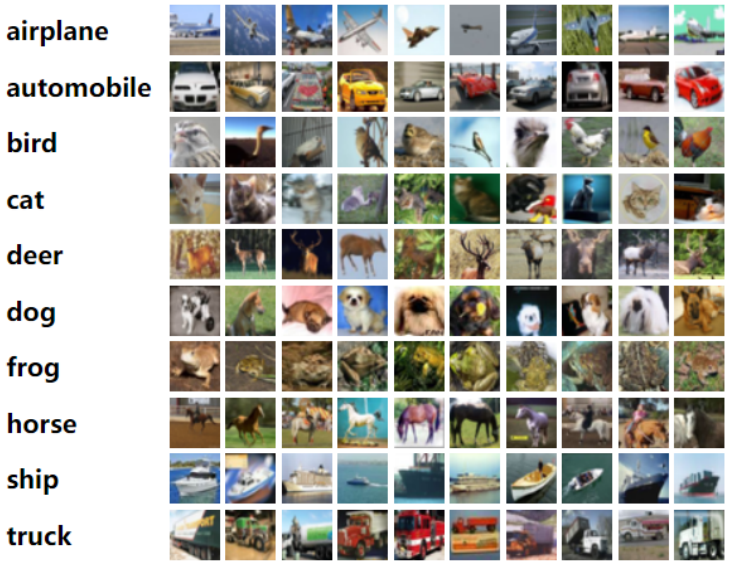
[8] Interpretability

تعریف مسئله

در این تمرین، در **بخش اول** به پیاده‌سازی یک شبکه‌ی کانولوشنی برای طبقه‌بندی تصاویر، با استفاده از کتابخانه Keras می‌پردازید. در **بخش دوم** و **سوم** اثر نامتوازن[9] بودن تعداد داده‌های آموزشی در کلاس‌های مختلف را بررسی می‌کنید و از تکنیک‌های Data Augmentation برای حل اثرات این موضوع استفاده می‌کنید. در بخش **آخر** نیز به بررسی روش‌های مختلف Regularization در فرآیند یادگیری و جلوگیری از overfitting می‌پردازید. برای راحتی استفاده از کتابخانه‌ها و تسریع فرآیند آموزش، می‌توانید از سرویس **Google Colab** استفاده کنید. توجه کنید که آموزش شبکه‌های عمیق (به خصوص شبکه‌های کانولوشنی) روی GPU بسیار سریع‌تر از CPU می‌باشد. در نتیجه می‌توانید Runtime Type سرویس Colab را روی GPU قرار دهید. دقت داشته باشید که مدت زمان استفاده از GPU سرویس Colab به ازای هر اکانت، محدود می‌باشد. در نتیجه مراحل پیش‌پردازش و تعریف مدل خود را روی CPU انجام داده و فقط حین آموزش مدل از GPU استفاده کنید تا مدت زمان استفاده از GPU اختصاصی شما به پایان نرسد.

معرفی مجموعه داده

مجموعه داده **CIFAR-10** مجموعه‌ای از تصاویر است که معمولاً برای آموزش الگوریتم‌های یادگیری ماشین و بینایی ماشین استفاده می‌شود. این دیتاست یکی از پرکاربردترین مجموعه‌های داده برای تحقیقات یادگیری ماشین است. مجموعه داده CIFAR-10 شامل ۶۰۰۰۰ تصویر رنگی ۳۲×۳۲ در ۱۰ کلاس مختلف است. ۱۰ کلاس مختلف هواپیما، ماشین، پرنده، گربه، آهو، سگ، قورباغه، اسب، کشتی و کامیون را نشان می‌دهد. از هر کلاس ۶۰۰۰ تصویر وجود دارد.

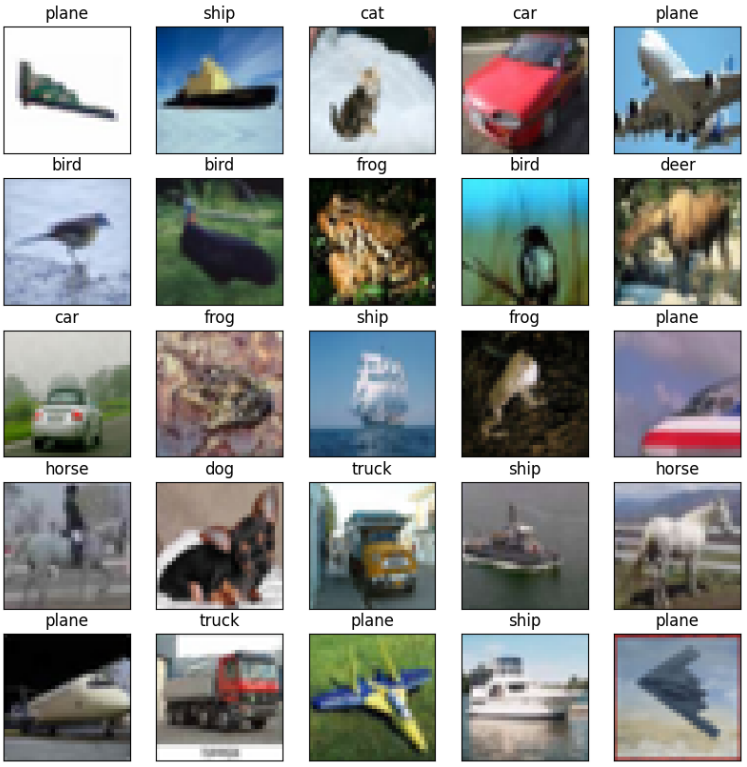


[9] Unbalanced Data

بخش صفر: آماده کردن داده

دیتاست CIFAR-10 را از tensorflow.keras.datasets لود کنید:

- یک تصویر از هر کلاس در دیتاست را نمایش دهید.
- تعداد داده‌های هر کلاس در مجموعه داده train و test را نمایش دهید.
- عملیات Normalization را برای کل تصاویر انجام دهید.
- برچسب‌های train و test را با استفاده از to_categorical به حالت OneHot تبدیل کنید.



شکل ۲. تعدادی از تصاویر مجموعه داده CIFAR-10

بخش اول: طبقه‌بندی تصاویر با استفاده از یک شبکه‌ی CNN

در بخش اول این پروژه، می‌خواهیم از یک شبکه کانولوشنی برای طبقه‌بندی تصاویر استفاده کنیم.

همانطور که توضیح داده شد، شبکه‌های کانولوشنی، با استفاده از لایه‌های کانولوشنی، لایه‌های ادغام و لایه‌های کاملا متصل، قادرند اطلاعات مربوط به الگوها و ویژگی‌های مختلف در تصویر را استخراج و استفاده کنند. لایه کانولوشنی با استفاده از عملیات کانولوشن، فیلترها را بر روی تصویر اعمال کرده و نمایش[10] جدیدی از تصویر را ایجاد می‌کند که شامل ویژگی‌های محلی است. این لایه‌ها به صورت مکرر در سراسر شبکه استفاده می‌شوند تا ویژگی‌های سطح بالاتر را استخراج کنند.

لایه‌های ادغام به منظور کاهش ابعاد تصویر و حذف اطلاعات بی‌اهمیت، استفاده می‌شوند. این لایه‌ها با استفاده از معیارهایی مانند حداکثرگیری[11] یا میانگین‌گیری[12] ، به کاهش ابعاد ویژگی‌های استخراج شده می‌پردازند.

لایه‌های کاملا متصل، در نهایت با استفاده از ویژگی‌های استخراج شده توسط لایه‌های کانولوشنی و ادغام، تصمیم‌گیری نهایی را برای طبقه‌بندی تصویر انجام می‌دهند. این لایه‌ها مشابه لایه‌های معمولی در شبکه‌های عصبی عمل می‌کنند و خروجی نهایی را تولید می‌کنند که شامل احتمال‌های مربوط به تعلق تصویر به هر کلاس است.

در ادامه یک نمونه کد برای تعریف یک شبکه کانولوشنی با استفاده از کتابخانه‌ی keras آمده است. توجه کنید که الزامی برای استفاده از این معماری خاص برای این بخش وجود ندارد و می‌توانید مدل زیر را تغییر داده و شبکه‌ی کانولوشنی خود را تعریف کنید.

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Activation, Input, Flatten
from keras.optimizers import Adam

cnn = Sequential()
cnn.add(Conv2D(32, (3, 3), padding='same', input_shape=x_train.shape[1:]))
cnn.add(Activation('relu'))
cnn.add(Conv2D(32, (3, 3)))
cnn.add(Activation('relu'))
cnn.add(MaxPooling2D(pool_size=(2, 2)))

cnn.add(Conv2D(64, (3, 3), padding='same'))
cnn.add(Activation('relu'))
cnn.add(Conv2D(64, (3, 3)))
cnn.add(Activation('relu'))
cnn.add(MaxPooling2D(pool_size=(2, 2)))

cnn.add(Flatten())
cnn.add(Dense(512))
```

[10] Representation

[11] Max Pooling

[12] Average Pooling

```
cnn.add(Activation('relu'))
cnn.add(Dense(10))
cnn.add(Activation('softmax'))

cnn.compile(loss = 'categorical_crossentropy',
            optimizer = 'adam',
            metrics = ['accuracy'])
log = cnn.fit(x_train, y_train_onehot, batch_size=32, epochs=20, validation_data=(x_test,
y_test_onehot))
```

در فاز اول پروژه با بهینه‌ساز SGD[13] آشنا شدید. این روش بهینه‌سازی با استفاده از روش گرادیان کاهشی به بهینه‌سازی وزن‌ها و پارامترهای شبکه‌ی عصبی می‌پردازد تا Loss کلی کاهش یافته و دقت افزایش یابد. بهینه‌سازهای قدرتمندتر دیگری نیز در کتابخانه keras وجود دارند. یکی از معروف‌ترین بهینه‌سازها، بهینه‌ساز **Adam** می‌باشد.

❓ **به طور مختصر نحوه کار بهینه‌ساز Adam و تفاوت آن با بهینه‌ساز SGD را توضیح دهید.**

- حال یک مدل Convolutional با استفاده از داده‌های train و Adam optimizer آموزش دهید و عملکرد شبکه را روی داده‌های test بررسی کنید و معیارهای Accuracy, Recall, Precision و F1-Score را گزارش کنید.

- **نکته ۱:** دقت کنید که ورودی شبکه‌ی عصبی در این فاز، تصاویر هستند که ساختار چند بعدی دارند. همانطور که در کد نمونه مشاهده می‌کنید، برای شبکه‌ی کانولوشنی، نیاز است که پس از اتمام لایه‌های کانولوشنی، ویژگی‌های استخراج شده توسط فیلترها Flatten شده و وارد لایه‌های Dense برای طبقه‌بندی شوند.
- **نکته ۲:** در تمامی مراحل تمرین، پس از اتمام تمامی epoch ها باید نمودار مقدار loss و accuracy بر حسب epoch را رسم کنید. برای راحتی کار در هر مرحله می‌توانید یک تابع برای این کار تعریف کرده و هر بار تابع خود را فراخوانی کنید.
- **نکته ۳:** در تمامی مراحل تمرین، پس از اتمام تمامی epoch ها باید معیارهای precision, recall, F1 برای داده‌های تمرین و تست گزارش شود[14].

[13] Stochastic Gradient Descent

[14] می‌توانید از تابع classification_report کتابخانه sklearn برای این منظور استفاده کنید.

بخش دوم: آموزش شبکه با داده‌های نامتوازن

وجود تعداد داده‌های نامتوازن می‌تواند باعث کاهش کارایی و دقت مدل در پیش‌بینی‌ها شود. وقتی داده‌ها در دسته‌بندی‌های مختلف توزیع نامتوازنی دارند، مدل تمایل به پیش‌بینی کلاس اکثریت خواهد داشت و نتایج اشتباهی برای کلاس‌های اقلیت تولید می‌کند.

برای روشن‌تر شدن، فرض کنید که در یک مسئله تشخیص تصویر، دو کلاس داریم: "گربه" و "سگ". اکنون فرض کنید که ۹۰٪ از تصاویر داده شده برای آموزش، تصاویر سگ است و ۱۰٪ تصاویر گربه. در این حالت، شبکه عصبی به سادگی می‌تواند همیشه کلاس سگ را پیش‌بینی کند و با این روش دقت ۹۰٪ را داشته باشد. اما واقعیت این است که شبکه عصبی وظیفه یادگیری ویژگی‌های هر دو کلاس را دارد و نمی‌تواند به درستی تمایز بین تصاویر گربه و سگ را تشخیص دهد.

برای مقابله با این نامتوازنی داده، روش‌های مختلفی وجود دارد. یک رویکرد متداول برای حل این مسئله استفاده از تکنیک‌های نمونه‌برداری مجدد[15] یا Data Augmentation است. در این روش، تعداد نمونه‌های کلاس اقلیت را افزایش می‌دهیم (از طریق تکرار نمونه‌های موجود یا تولید نمونه‌های مصنوعی مشابه نمونه‌های اصلی) تا تعادلی بین کلاس‌ها برقرار شود.

همچنین، معیارهای مختلفی مانند AUC (مساحت زیر نمودار ROC) و ماتریس درهم‌ریختگی[16] می‌توانند برای ارزیابی صحت مدل در حالت نامتوازن مورد استفاده قرار گیرند که در این پروژه مورد توجه ما نمی‌باشند. این معیارها می‌توانند برای نشان دادن کارایی و عملکرد مدل در تشخیص کلاس‌های کمتر و بررسی نقاط ضعف و قوت مدل در مواجهه با داده‌های نامتوازن مورد استفاده قرار بگیرند.

در این بخش به بررسی تاثیر وجود تعداد داده‌های نامتوازن در مجموعه داده خود می‌پردازیم.

- ابتدا ۸۰ درصد داده‌های کلاس اول و دوم (یعنی هواپیما و ماشین) را به صورت تصادفی انتخاب کرده و از مجموعه داده‌ی آموزش خود حذف کنید (داده‌ی تست بدون تغییر باقی می‌ماند).
- شبکه‌ی کانولوشنی مشابه مدلی که در بخش اول استفاده کردید، تعریف کنید و این شبکه را روی داده‌های نامتوازنی که ایجاد کردید آموزش دهید.
- نتایج Recall و F-1 روی داده‌ی تست کلاس هواپیما و ماشین را با **بخش اول** مقایسه کنید. دلیل تغییر این مقادیر را شرح دهید.

---

[15] re-sampling

[16] Confusion Matrix

بخش سوم: استفاده از Data Augmentation برای ایجاد توازن بین تعداد داده‌ها

همانطور که اشاره شد، یک راه برای مقابله با مشکل تعداد داده‌های نامتوازن در مجموعه داده، استفاده از روش‌های تقویت داده یا Data Augmentation است. این روش‌ها برای افزایش تنوع داده‌ها و ایجاد تعادل بین کلاس‌ها استفاده می‌شوند.

یکی از روش‌های متداول برای افزایش تعداد داده‌ها در شبکه‌های عصبی کانولوشنی، اعمال تبدیلات جزئی بر روی تصاویر است. مثلاً می‌توان تصاویر را به صورت تصادفی چرخاند، اندازه‌ی آن‌ها را تغییر داد، آنها را به صورت تصادفی برش داد، شدت روشنایی را تغییر داد و یا با اعمال نویز به تصاویر، تنوع بیشتری از تصاویر موجود به دست آورد. بدین ترتیب شبکه عصبی می‌تواند الگوهای مختلف را یاد بگیرد و مشکل نامتوازن بودن تعداد داده‌های کلاس‌های مختلف تا حدی برطرف شود.

در این بخش، باید این تبدیلات را به صورت تصادفی بر روی تصاویر مجموعه داده‌ی آموزش اعمال کنید. با این کار، تعداد داده‌های هواپیما و ماشین را متعادل‌تر می‌کنیم و شبکه عصبی قادر خواهد بود تا الگوهای هر دو کلاس را به خوبی یاد بگیرد.

نکته مهمی که باید در نظر داشته باشیم، این است که Data Augmentation را فقط بر روی داده‌های آموزش اعمال می‌کنیم و داده‌های تست را بدون هیچگونه تغییری باقی می‌گذاریم. در واقع، ما فقط تصاویر اصلی را در مجموعه داده‌ی آموزش داریم و تبدیلات تصادفی را هر بار بر روی تصاویر اعمال می‌کنیم تا داده‌های تقویت شده‌ای بسازیم.

- با استفاده از روش‌هایی مانند RandomZoom، RandomFlip، RandomRotation یا هر روش دیگری که به نظر شما مناسب است، تعداد داده‌های کلاس هواپیما و ماشین را افزایش داده و دوباره مجموعه داده‌ای متوازن ایجاد کنید.
- شبکه‌ی کانولوشنی مشابه مدلی که در بخش اول و دوم استفاده کردید، تعریف کنید و این شبکه را روی داده‌های تقویت شده‌ای که ایجاد کردید آموزش دهید.
- نتایج Recall و F-1 روی داده‌ی تست کلاس هواپیما و ماشین را با **بخش دوم** مقایسه کنید. نتایج به دست آمده را تحلیل کنید.

بخش چهارم: تاثیر روش‌های Regularization در فرآیند آموزش

روش‌های Regularization در فرآیند آموزش شبکه عصبی مورد استفاده قرار می‌گیرند تا از بیش‌برازش[17] جلوگیری کنند. دو روش معروف در این زمینه Dropout و Batch Normalization هستند که به ترتیب در ادامه توضیح داده خواهند شد:

1. Dropout: این روش یکی از تکنیک‌های مؤثر برای کاهش بیش‌برازش در شبکه‌های عصبی است. این روش در فرآیند آموزش، به صورت تصادفی برخی از نورون‌ها را در هر مرحله غیرفعال می‌کند. به این ترتیب، هر نورون مجبور است الگوهای مفید را در حضور نورون‌های دیگر یاد بگیرد و به این ترتیب میزان وابستگی نورون‌ها به هم و همچنین میزان وابستگی شبکه به هر نورون کاهش می‌یابد. این باعث می‌شود که شبکه‌ی عصبی توانایی تعمیم‌پذیری بهتری پیدا کند و کمتر دچار بیش‌برازش شود. Dropout می‌تواند به عنوان یک لایه Dropout با ضریب احتمال Dropout خاصی در ساختار شبکه اعمال شود.
2. Batch Normalization: این تکنیک، یک روش استانداردسازی ورودی‌های هر بچ (batch) در یک لایه است. در هر بچ از داده‌ها، میانگین و واریانس ورودی‌ها محاسبه شده و سپس ورودی‌ها به صورت استانداردسازی شده با استفاده از میانگین و واریانس محاسبه شده، وارد لایه‌ی بعدی می‌شوند. این روش باعث می‌شود که توزیع ورودی‌ها در هر لایه بهبود یابد و بیش‌برازش کاهش یابد. علاوه بر این، Batch Normalization نیز می‌تواند به عنوان یک لایه BatchNorm در ساختار شبکه اعمال شود.

در نتیجه، استفاده از Dropout و Batch Normalization در شبکه عصبی، می‌تواند تأثیر مثبتی بر عملکرد شبکه در فرآیند آموزش داشته باشد. این روش‌ها موجب کاهش بیش‌برازش، بهبود تعمیم‌پذیری و عملکرد مدل در مجموعه داده‌های تست می‌شوند. با اعمال این روش‌ها، شبکه عصبی قادر خواهد بود الگوهای مفید را یاد بگیرد و همچنین مقاومت بیشتری در برابر داده‌های نویزی و تغییرات کوچک در ورودی‌ها خواهد داشت.

در این بخش، باید از لایه Dropout بین لایه‌های Dense و لایه Batch Normalization بین لایه‌های کانولوشنی در شبکه خود استفاده کنید.

- شبکه‌ی کانولوشنی مشابه مدل‌های قبلی، تعریف کنید و لایه‌های Dropout و BatchNorm را با توجه به توضیحات ارائه شده به مدل خود اضافه کنید. این شبکه را روی داده‌های اصلی آموزش دهید (از داده‌های نامتوازن یا augment شده در این بخش استفاده نکنید).
- نتایج Recall و F-1 روی داده‌ی تست را با **بخش اول** مقایسه کنید. نتایج به دست آمده را تحلیل کنید.

[17] overfitting

منابع

برای یادگیری شروع کار با Keras میتوانید از این لینک کمک بگیرید.

برای آموزش شیوه استفاده از Google Colab می‌توانید از این لینک یا این لینک استفاده نمایید.

در این لینک درباره‌ی Adam Optimizer مطالعه کنید.

در این لینک درباره‌ی روش‌های مختلف regularization مطالعه کنید.

نکات پایانی

- دقت کنید که هدف پروژه **تحلیل نتایج** و تاثیر عوامل مختلف است؛ بنابراین از ابزارهای تحلیل داده بطور مثال نمودارها استفاده کنید و توضیحات مربوط به هر بخش از پروژه را بطور خلاصه و در عین حال مفید در گزارش خود ذکر کنید.
- نتایج و گزارش خود را در یک فایل فشرده با عنوان AI-CA5-P2-<#SID>.zip تحویل دهید. محتویات پوشه باید شامل فایل jupyter-notebook، خروجی html و فایل‌های مورد نیاز برای اجرای آن باشد. **تحلیل و نمایش خروجی‌های خواسته شده بخشی از نمره این تمرین را تشکیل می‌دهد.** از نمایش درست خروجی‌های مورد نیاز در فایل html مطمئن شوید.
- توجه داشته باشید که علاوه بر ارسال فایل‌های پروژه، این پروژه به صورت حضوری نیز تحویل گرفته خواهد شد. بنابراین تمام بخش‌های پروژه باید قابلیت اجرای مجدد در زمان تحویل حضوری را داشته باشند. همچنین در صورت عدم حضور در تحویل حضوری نمره‌ای دریافت نخواهید کرد.
- در صورتی که سوالی در مورد پروژه داشتید بهتر است در فروم یا گروه درس مطرح کنید تا بقیه از آن استفاده کنند؛ در غیر این صورت با طراحان در ارتباط باشید.
- هدف از تمرین، یادگیری شماست. لطفا تمرین را خودتان انجام دهید.

موفق باشید.