**فصل ۸ - مقدمه‌ای بر یادگیری عمیق برای بینایی کامپیوتری**

چرا نباید لایه‌های بیشتری تنظیم دقیق شوند؟ چرا نباید کل شبکه کانولوشنی تنظیم دقیق شود؟

• شما می‌توانید این کار را انجام دهید، اما باید نکات زیر را در نظر بگیرید:

1. لایه‌های اولیه در شبکه کانولوشنی ویژگی‌های عمومی و قابل استفاده مجدد را رمزگذاری می‌کنند، در حالی که لایه‌های بالاتر ویژگی‌های تخصصی‌تری دارند.

2. تنظیم دقیق لایه‌های بالاتر مفیدتر است، زیرا این ویژگی‌ها به بازنگری بیشتری برای حل مسئله جدید نیاز دارند.

3. ریسک بیش‌برازش (Overfitting) با افزایش تعداد پارامترهای قابل آموزش بیشتر می‌شود. شبکه کانولوشنی دارای ۱۵ میلیون پارامتر است که آموزش آن با مجموعه داده کوچک، ریسک بالایی دارد.

پیاده‌سازی تنظیم دقیق مدل

conv\_base.trainable = True

for layer in conv\_base.layers[:-4]:

layer.trainable = False

کامپایل و آموزش مدل:

model.compile(loss="binary\_crossentropy",

optimizer=keras.optimizers.RMSprop(learning\_rate=1e-5),

metrics=["accuracy"])

callbacks = [

keras.callbacks.ModelCheckpoint(

filepath="fine\_tuning.keras",

save\_best\_only=True,

monitor="val\_loss")

]

history = model.fit(

train\_dataset,

epochs=30,

validation\_data=validation\_dataset,

callbacks=callbacks)

ارزیابی مدل:

model = keras.models.load\_model("fine\_tuning.keras")

test\_loss, test\_acc = model.evaluate(test\_dataset)

print(f"Test accuracy: {test\_acc:.3f}")

خلاصه:

1. شبکه‌های کانولوشنی (ConvNets) بهترین مدل‌ها برای وظایف بینایی کامپیوتری هستند.

2. برای مجموعه داده کوچک، بیش‌برازش مشکل اصلی است. افزایش داده‌ها (Data Augmentation) یک روش قدرتمند برای مقابله با آن است.

3. استخراج ویژگی‌ها از شبکه‌های از پیش آموزش‌دیده‌شده یک تکنیک ارزشمند برای مجموعه داده‌های کوچک است.

4. تنظیم دقیق می‌تواند عملکرد مدل را بیشتر بهبود بخشد.

فصل ۹ - یادگیری عمیق پیشرفته برای بینایی کامپیوتری

سه وظیفه اصلی بینایی کامپیوتری:

1. طبقه‌بندی تصویر (Image Classification) - تخصیص یک یا چند برچسب به تصویر.

2. تقسیم‌بندی تصویر (Image Segmentation) - تقسیم تصویر به بخش‌های مختلف بر اساس دسته‌بندی‌ها.

3. تشخیص اشیا (Object Detection) - رسم کادرهای مرزبندی (Bounding Boxes) به دور اشیای مورد نظر و اختصاص کلاس‌ها به آن‌ها.

توضیح تقسیم‌بندی تصویر:

تقسیم‌بندی تصویر به دو نوع اصلی تقسیم می‌شود:

• تقسیم‌بندی معنایی (Semantic Segmentation): دسته‌بندی هر پیکسل به یک دسته خاص، مانند «گربه».

• تقسیم‌بندی نمونه‌ای (Instance Segmentation): تفکیک اشیا به نمونه‌های جداگانه مانند «گربه ۱» و «گربه ۲».

مثال پیاده‌سازی تقسیم‌بندی تصویر با مجموعه داده Oxford-IIIT Pets:

۱. دریافت و آماده‌سازی داده‌ها:

!wget http:/ /www.robots.ox.ac.uk/~vgg/data/pets/data/images.tar.gz

!wget http:/ /www.robots.ox.ac.uk/~vgg/data/pets/data/annotations.tar.gz

!tar -xf images.tar.gz

!tar -xf annotations.tar.gz

۲. پردازش مسیر فایل‌ها:

import os

input\_dir = "images/"

target\_dir = "annotations/trimaps/"

input\_img\_paths = sorted(

[os.path.join(input\_dir, fname)

for fname in os.listdir(input\_dir)

if fname.endswith(".jpg")])

target\_paths = sorted(

[os.path.join(target\_dir, fname)

for fname in os.listdir(target\_dir)

if fname.endswith(".png") and not fname.startswith(".")])

۳. نمایش نمونه‌ای از تصاویر و ماسک‌ها

import matplotlib.pyplot as plt

from tensorflow.keras.utils import load\_img, img\_to\_array

plt.axis("off")

plt.imshow(load\_img(input\_img\_paths[9]))

۴. آماده‌سازی داده‌ها برای آموزش:

import numpy as np

import random

img\_size = (200, 200)

num\_imgs = len(input\_img\_paths)

random.Random(1337).shuffle(input\_img\_paths)

random.Random(1337).shuffle(target\_paths)

input\_imgs = np.zeros((num\_imgs,) + img\_size + (3,), dtype="float32")

targets = np.zeros((num\_imgs,) + img\_size + (1,), dtype="uint8")

for i in range(num\_imgs):

input\_imgs[i] = img\_to\_array(load\_img(input\_img\_paths[i], target\_size=img\_size))

targets[i] = img\_to\_array(load\_img(target\_paths[i], target\_size=img\_size, color\_mode="grayscale")).astype("uint8") - 1

train\_input\_imgs = input\_imgs[:-1000]

train\_targets = targets[:-1000]

val\_input\_imgs = input\_imgs[-1000:]

val\_targets = targets[-1000:]

۵. تعریف مدل تقسیم‌بندی:

from tensorflow import keras

from tensorflow.keras import layers

def get\_model(img\_size, num\_classes):

inputs = keras.Input(shape=img\_size + (3,))

x = layers.Rescaling(1./255)(inputs)

x = layers.Conv2D(64, 3, strides=2, activation="relu", padding="same")(x)

x = layers.Conv2D(64, 3, activation="relu", padding="same")(x)

x = layers.Conv2D(128, 3, strides=2, activation="relu", padding="same")(x)

x = layers.Conv2DTranspose(128, 3, strides=2, activation="relu", padding="same")(x)

x = layers.Conv2DTranspose(64, 3, strides=2, activation="relu", padding="same")(x)

outputs = layers.Conv2D(num\_classes, 3, activation="softmax", padding="same")(x)

model = keras.Model(inputs, outputs)

return model

model = get\_model(img\_size=img\_size, num\_classes=3)

model.summary()

۶. خلاصه مدل:

مدل شامل لایه‌های کانولوشنی (Conv2D) و بازگردانی کانولوشنی (Conv2DTranspose) برای استخراج ویژگی‌ها و بازسازی تصاویر است.

- **فصل ۹: یادگیری عمیق پیشرفته برای بینایی کامپیوتری**

ساختار مدل و تحلیل عملکرد آن

مدل طراحی شده شامل دو بخش اصلی است:

1. بخش اول (Encoding):

• شامل لایه‌های کانولوشنی (Conv2D) است که به تدریج اندازه تصویر را کاهش داده و ویژگی‌ها را استخراج می‌کند.

• این لایه‌ها اطلاعات مکانی تصویر را فشرده می‌کنند و نمایش‌های سطح بالاتری از ویژگی‌ها ایجاد می‌کنند.

2. بخش دوم (Decoding):

• شامل لایه‌های بازگردانی کانولوشنی (Conv2DTranspose) است که اندازه تصویر را به تدریج افزایش داده و ویژگی‌های استخراج‌شده را بازسازی می‌کند.

• هدف این بخش ایجاد خروجی نهایی است که هر پیکسل را به یک کلاس خاص (مانند پس‌زمینه، پیش‌زمینه یا مرز) اختصاص دهد.

بررسی اجزای مدل:

Layer (type) Output Shape Param #

=================================================================

input\_1 (InputLayer) [(None, 200, 200, 3)] 0

rescaling (Rescaling) (None, 200, 200, 3) 0

conv2d (Conv2D) (None, 100, 100, 64) 1792

conv2d\_1 (Conv2D) (None, 100, 100, 64) 36928

conv2d\_2 (Conv2D) (None, 50, 50, 128) 73856

conv2d\_3 (Conv2D) (None, 50, 50, 128) 147584

conv2d\_transpose (Conv2DTr) (None, 100, 100, 64) 36928

conv2d\_4 (Conv2D) (None, 200, 200, 3) 1731

=================================================================

Total params: 512,819

Trainable params: 512,819

تحلیل مدل:

• بخش اول تصویر را سه بار کوچک‌تر کرده و ویژگی‌ها را فشرده می‌کند.

• بخش دوم ویژگی‌ها را بازسازی می‌کند و پیکسل‌ها را به دسته‌های مربوطه تقسیم‌بندی می‌کند.

• در این مدل از لایه‌های با گام‌برداری (Strides) به جای لایه‌های بیشینه‌برداری (MaxPooling) استفاده شده است. دلیل این انتخاب، حفظ اطلاعات مکانی برای تولید ماسک‌های دقیق پیکسلی است.

آموزش مدل:

برای آموزش مدل، ابتدا باید آن را کامپایل کرده و یک تابع زیان (Loss Function) مناسب انتخاب شود.

۱. کامپایل و آموزش مدل:

model.compile(

optimizer="adam",

loss="sparse\_categorical\_crossentropy",

metrics=["accuracy"])

history = model.fit(

train\_input\_imgs,

train\_targets,

batch\_size=32,

epochs=15,

validation\_data=(val\_input\_imgs, val\_targets))

۲. نمایش روند آموزش:

import matplotlib.pyplot as plt

plt.plot(history.history["accuracy"], label="Accuracy (Train)")

plt.plot(history.history["val\_accuracy"], label="Accuracy (Validation)")

plt.xlabel("Epoch")

plt.ylabel("Accuracy")

plt.legend()

plt.show()

۳. پیش‌بینی و ارزیابی مدل:

predictions = model.predict(val\_input\_imgs)

plt.imshow(val\_input\_imgs[0].astype("uint8"))

plt.show()

plt.imshow(np.argmax(predictions[0], axis=-1))

plt.show()

نکات مهم درباره مدل:

1. مدل برای تقسیم‌بندی معنایی (Semantic Segmentation) طراحی شده است.

2. ماسک خروجی شامل دسته‌بندی هر پیکسل به عنوان پیش‌زمینه، پس‌زمینه یا مرز است.

3. استفاده از تابع زیان «Categorical Crossentropy» برای پردازش چندکلاسه پیکسل‌ها.

نتایج مدل:

1. دقت بالا: مدل قادر به دستیابی به دقتی بالاتر از ۹۵٪ در مجموعه داده اعتبارسنجی است.

2. عملکرد سریع: مدل با مجموعه داده‌های کوچک به سرعت آموزش می‌بیند.

3. کاربردهای متنوع: این مدل برای کاربردهایی مانند ویرایش تصویر، رانندگی خودکار و تشخیص پزشکی قابل استفاده است.

جمع‌بندی:

این فایل آموزشی به بررسی تکنیک‌های یادگیری عمیق برای بینایی کامپیوتری پرداخت.

• در ابتدا با مدل‌های ساده برای طبقه‌بندی تصاویر شروع شد.

• سپس به مباحث پیشرفته‌تر مانند تقسیم‌بندی تصاویر پرداخته شد.

• همچنین کدهای نمونه برای پیاده‌سازی، آموزش و ارزیابی مدل‌ها ارائه شد.

تکنیک‌های کلیدی معرفی‌شده:

1. تنظیم دقیق مدل‌ها برای داده‌های جدید.

2. استفاده از استخراج ویژگی‌ها برای مجموعه داده‌های کوچک.

3. طراحی مدل‌های تقسیم‌بندی معنایی با قابلیت بازسازی ماسک‌های دقیق.

**خلاصه ای از ترجمه**

**مقدمه**

**در این متن، روش‌های یادگیری عمیق برای بینایی کامپیوتر توضیح داده شده است. این روش‌ها شامل مواردی مانند تشخیص تصویر، تقسیم‌بندی تصویر، و تشخیص اشیا هستند. هدف اصلی این است که با استفاده از مدل‌های یادگیری عمیق، اطلاعات تصویری به داده‌های قابل تحلیل تبدیل شوند.**

**بخش اول: مفهوم Fine-Tuning یا تنظیم مجدد مدل**

**Fine-Tuning چیست؟**

**این تکنیک شامل بهبود مدل‌های از قبل آموزش‌داده‌شده برای مسائل جدید است.**

**چرا همه لایه‌ها تنظیم نمی‌شوند؟**

**• لایه‌های پایین‌تر: ویژگی‌های عمومی‌تر را یاد می‌گیرند که در مسائل مختلف قابل استفاده هستند.**

**• لایه‌های بالاتر: ویژگی‌های تخصصی‌تر مرتبط با مسئله اصلی را یاد می‌گیرند.**

**نکته کلیدی: بهتر است فقط لایه‌های بالایی مدل تغییر داده شوند تا از پیچیدگی و خطرات «Overfitting» (جور شدن بیش از حد با داده‌ها) جلوگیری شود.**

**بخش دوم: سه وظیفه اصلی در بینایی کامپیوتر**

**۱. تشخیص تصویر: شناسایی موضوع اصلی در تصویر (مثلاً گربه یا سگ).**

**۲. تقسیم‌بندی تصویر: جداسازی بخش‌های مختلف تصویر، مثلاً پس‌زمینه و موضوع اصلی.**

**۳. تشخیص اشیا: مشخص کردن مکان و نوع اشیای مختلف در تصویر.**

**بخش سوم: یک مثال از تقسیم‌بندی تصویر**

**تقسیم‌بندی تصویر چیست؟**

**این فرآیند شامل طبقه‌بندی هر پیکسل تصویر به دسته‌بندی‌های مختلف است. دو نوع اصلی تقسیم‌بندی:**

**• تقسیم‌بندی معنایی: هر پیکسل براساس مفهوم دسته‌بندی می‌شود (مثل “گربه”).**

**• تقسیم‌بندی موردی: هر شیء مجزا (مثل “گربه ۱” و “گربه ۲”) جداگانه شناسایی می‌شود.**

**نمونه استفاده:**

**در پروژه‌ای، داده‌های مربوط به گربه‌ها و سگ‌ها به همراه ماسک‌های تقسیم‌بندی (نقشه‌ای از پیکسل‌ها که هر پیکسل را به یک دسته تخصیص می‌دهد) تحلیل می‌شوند.**

**بخش چهارم: مدل‌سازی با شبکه‌های عصبی**

**مدل‌های یادگیری عمیق شامل دو بخش اصلی هستند:**

**• بخش اول: کاهش ابعاد تصویر و استخراج ویژگی‌ها.**

**• بخش دوم: بازسازی اطلاعات با استفاده از لایه‌های تخصصی (مانند لایه‌های Conv2DTranspose).**

**نکات مهم:**

**• از تکنیک‌هایی استفاده می‌شود که اطلاعات مکانی (Spatial Information) پیکسل‌ها حفظ شوند.**

**• به‌جای استفاده از روش‌های سنتی مانند MaxPooling، از «Convolution Strides» برای کاهش اندازه استفاده می‌شود.**