دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

گزارش تمرین هوش سایبری

Not Really simple Captcha

پدیدآورنده :

محمد امین کیانی ۴۰۰۳۶۱۳۰۵۲

دانشجوی کارشناسی، دانشکده‌ی کامپیوتر، دانشگاه اصفهان، اصفهان،
Aminkianiworkeng@gmail.com

استاد راهنما: جناب اقای دکتر مهدوی

نیمسال دوم تحصیلی ۰۳-۱۴۰۲

فهرست مطالب

مستندات

۱-مسئله و تحلیل کلی آن:

پیاده سازی یک الگوریتم یا بدسته ابزار برای شکستن کپچاهای خط خطی است. این کپچاها اغلب شامل کاراکترهای عددی و حروفی به صورت خط خطی شده هستند.

۱. تشخیص و تصحیح کپچاها به صورت اتوماتیک

۲. ساخت یک سیستم یا الگوریتم برای شکستن کپچاهای پیچیده

۳. امکان بهبود عملکرد الگوریتم برای مواجهه با چالش‌های مختلف در کپچا

شاخص های عملکرد:

- دقت تشخیص و تصحیح کپچاها

- زمان لازم برای شکستن کپچاها

- کارایی الگوریتم در مواجهه با چالش‌های پیچیده کپچا

تشخیص کپچا (Captcha Recognition Project) یک عمل گرافیک محاسباتی است که به منظور تشخیص و حل کدهای Captcha کاربرد دارد. Captcha یک ابزار امنیتی است که بعضی وبسایت‌ها از آن برای جلوگیری از حملات رباتیک و اسپم استفاده می‌کنند.

ابتدا باید یک مجموعه داده از تصاویر Captcha جمع‌آوری کرد. سپس، الگوریتم‌های یادگیری ماشین یا شبکه‌های عصبی برای آموزش به تشخیص کدهای Captcha بر روی این داده‌ها استفاده می‌شود.

مراحل کلی به شرح زیر است:

۱. جمع‌آوری داده‌های Captcha

۲. پیش‌پردازش تصاویر (تبدیل به سیاه و سفید، اندازه‌بندی، حذف نویز و ...)

۳. آموزش الگوریتم‌های یادگیری ماشین یا شبکه‌های عصبی

۴. ارزیابی عملکرد سیستم بر روی داده‌های تست

۵. بهینه‌سازی و افزایش دقت تشخیص

در نهایت، پس از آموزش موفق سیستم، می‌توان از آن برای تشخیص کدهای Captcha در وبسایت‌ها و جلوگیری از حملات اسپم و رباتیک استفاده کرد.

۲-تغییر بعد:

```
def resize_image_to_dimensions(image, desired_width, desired_height):
    """Resizes an image to the desired dimensions."""
    (h, w) = image.shape[:2]
    if w > h:
        image = imutils.resize(image, width=desired_width)
    else:
        image = imutils.resize(image, height=desired_height)
    pad_width = int((desired_width - image.shape[1]) / 2.0)
    pad_height = int((desired_height - image.shape[0]) / 2.0)
    image_with_border = cv2.copyMakeBorder(
        image, pad_height, pad_height, pad_width, pad_width, cv2.BORDER_REPLICATE
    )
    image_with_border_resized = cv2.resize(
        image_with_border, (desired_width, desired_height)
    )
    return image_with_border_resized
```

برای انجام این کار، کتابخانه OpenCV برای پردازش تصویر استفاده شده است.

این بخش یک تصویر ورودی را به ابعاد دلخواه تغییر اندازه می‌دهد. برای انجام این‌کار، کد ابتدا اندازه اولیه تصویر را بررسی می‌کند و سپس با توجه به اینکه عرض تصویر بیشتر است یا ارتفاع، تصویر را به اندازه دلخواه تغییر اندازه می‌دهد. بعد از تغییر اندازه، کد وسط‌کردن تصویر را انجام می‌دهد تا تصویر ورودی در وسط قرار بگیرد. اینکار با تعیین یک میزان پادینگ در افقی و عمودی به دست می‌آید. در نهایت، تصویر وسط‌کرده شده به ابعاد دلخواه برگردانده می‌شود.

۳-انتخاب مدل و لایه بندی بدون Dropout :

```
from keras.models import Sequential
from keras.layers.convolutional import Conv2D,MaxPooling2D
from keras.layers.core import Dense,Flatten


num_classes = 32
NN_model = Sequential()
NN_model.add(
    Conv2D(20, (5, 5), padding="same", input_shape=(20, 20, 1),
activation="relu")
)
NN_model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
NN_model.add(Conv2D(50, (5, 5), padding="same", activation="relu"))
NN_model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
NN_model.add(Flatten())
NN_model.add(Dense(512, activation="relu"))
NN_model.add(Dense(num_classes, activation="softmax"))
NN_model.compile(
    loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"]
)
NN_model.summary()
```

در اینجا، از TensorFlow برای تعریف این معماری استفاده شده است.

سپس نیاز به افزودن لایه‌های لایه‌ها و تنظیمات مدل، مانند تعداد نورون‌ها و توابع فعال‌سازی مربوطه، خواهیم داشت. این مدل می‌تواند در پیش‌بینی یا دسته‌بندی ارقام دست نوشته شده در تصاویر به کار رود.

به طور کلی، اضافه کردن لایه‌های عمیق‌تر می‌تواند به مدل قدرت بیشتری برای یادگیری اطلاعات پیچیده بدهد اما نیاز به مراقبت بیشتر در مورد تطبیق آن با داده‌ها و جلوگیری از overfitting وجود دارد. پس با تست گذاشتن چندین حالت مختلف از تعداد لایه های متفاوت و تحقیق در کتاب هندز ان ماشین لرنینگ میتوان نتیجه گرفت که این میزان لایه برای مدل انتخاب شده توسط بنده کافی است و از اورفیت شدن هم جلو گیری می کند و لایه های بیشتر از این نیاز نبوده و تنها سرعت اجرا را پایین می برند و لایه ی کمتر نیز دقت و قدرت را پایین می برد.

tf.keras.layers.BatchNormalization() در TensorFlow یک لایه‌ی Batch Normalization اضافه می‌کند تا آموزش شبکه عصبی سریعتر شود، مشکل محوشوندگی را کاهش می‌دهد، باعث افزایش دقت مدل خود می‌شود و از مشکل برازش بیش‌افرازی جلوگیری می‌کند. این لایه به آموزش شبکه عصبی کمک می‌کند تا به سرعت به یک حالت تعادل مطلوب برسد. که در اینجا استفاده نشده است!

لایه Dropout به طور تصادفی بخشی از ورودی‌ها را با احتمالی حذف می‌کند. این کار باعث کاهش اورفیت مدل می‌شود و از بروز پدیده‌هایی مانند بیش‌برازش جلوگیری می‌کند. که در اینجا استفاده نشده است!

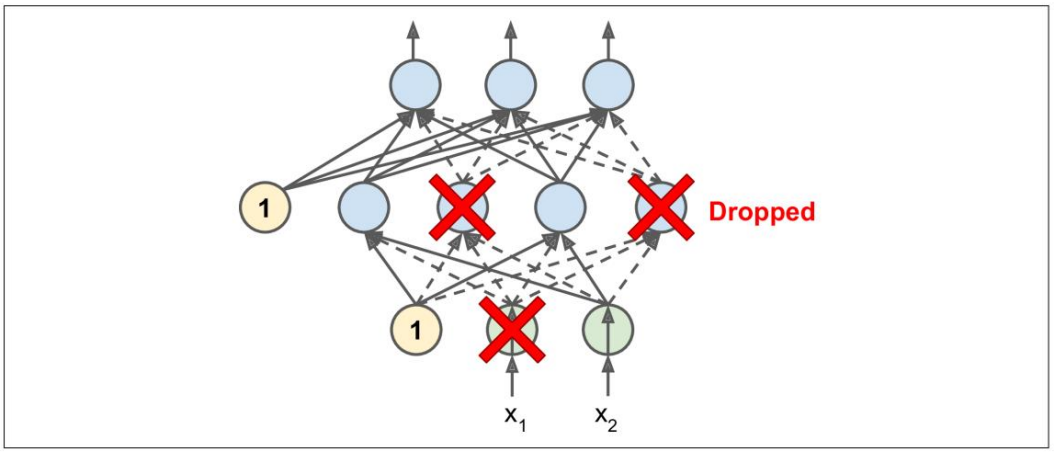

Figure 11-9. Dropout regularization

٤-تاثیر توابع فعالسازی:

ReLU (Rectified Linear Activation): این تابع غیرخطی است و در لایه‌های مخفی شبکه عصبی به طور گسترده استفاده می‌شود. این تابع اعداد منفی را به صفر تبدیل می‌کند.

استفاده از تابع فعال‌سازی Relu (Rectified Linear Unit) به طور معمول در شبکه‌های عصبی عمیق (Deep Neural Networks) توصیه شده است، زیرا این تابع بهبود مهمی در آموزش و عملکرد مدل‌ها می‌آورد. از مزایای استفاده از Relu می‌توان به سرعت آموزش، جلوگیری از مشکل مواجهه با مشکل مرگ نورون (Vanishing Gradient Problem) و افزایش قدرت انتقال سیگنال‌های غیرخطی اشاره کرد. به همین دلیل استفاده از Relu به جای Sigmoid در شبکه‌های عصبی رایج تر است. در واقع سیگموید را میتوان با سافت مکس مقایسه کرد که در ادامه شکست ان را میبینیم زیرا برای مثال دو کلاسه خوب است!

Softmax: این تابع بیشتر برای مسائل طبقه‌بندی استفاده می‌شود. این تابع ورودی‌های خروجی را به احتمالات مقابله‌ای تبدیل می‌کند که مجموع آن‌ها برابر با ۱ است، بنابراین می‌توان احتمال تعلق هر ورودی به هر کلاس را مشخص کرد.

تابع sigmoid به عنوان تابع فعال‌ساز در مسائل دسته‌بندی دو دسته‌ای (binary classification) معمولاً استفاده می‌شود، زیرا اعداد را به بازه ۰ تا ۱ محدود می‌کند که متناسب با خروجی‌های تنها یک برچسب است.

۵-نرخ یادگیری و الگوریتم های بهینه سازی:

در مدل‌های شبکه‌های عصبی، نرخ یادگیری (learning rate) میزانی است که مشخص می‌کند که چقدر وزن‌های شبکه در هر مرحله به سمت جواب بهینه تغییر کنند. این نرخ یادگیری می تواند بر اساس تجربه و تلاش های انجام شده توسط افراد و یا با استفاده از الگوریتم‌های بهینه‌سازی مشخص شود.

برای کد تایید نرخ یادگیری در مدل‌های شبکه‌های عصبی، معمولا از رویکردهای زیر استفاده می‌شود:

۱. Grid Search: با استفاده از روش Grid Search، می‌توان یک مجموعه از مقادیر نرخ یادگیری را تعیین کرده و سپس مدل را بر اساس هرکدام از این مقادیر آموزش داده و به دنبال بهترین عملکرد مدل با توجه به مقدار نرخ یادگیری باشیم.

۲. Random Search: در این روش، مقادیر نرخ یادگیری به صورت تصادفی انتخاب می‌شوند و مدل بر اساس این مقادیر آموزش داده می‌شود. این روش می‌تواند به صورت موثرتری مقدار بهینه را پیدا کند به واسطه جستجو در فضای مقادیر به صورت تصادفی.

۳. Optimization Algorithms: الگوریتم‌های بهینه‌سازی مانند Adam, RMSprop و SGD می‌توانند کمک کنند تا نرخ یادگیری بهینه برای مدل شبکه عصبی شما پیدا شود. که در اینجا ما از ادام برای بهبود نرخ یادگیر استفاده کردیم. که البته ادام به صورت دیفالت اگر نرخ یادگیری برایش تعیین نکنیم دیفالت ۰.۰۰۱ میگذارد یعنی :

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
==========================================================================#
NN_model.compile(
    loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"]
)
```

حال با تغییر این عدد و تست مقادیر مختلف مثلا ۰.۰۰۰۰۰۱=learning_rate باعث می شد دقت پایین تر رود درواقع در اکسترمم های محلی گیر کند و گام برداشتن ان بسیار کوچک باشد. از طرفی هم بزرگ تر کردن ان سبب دور شدن ناگهانی از اکسترمم گلوبال بود و نمودار ها را از فیت دور تر می کرد:

بهینه‌سازی‌ها در اصل الگوریتم‌هایی هستند که هدف آن‌ها بهینه کردن پارامترهای مدل به نحوی است که تابع هدف کاهش یابد. بهینه‌سازها می‌توانند با تنظیم نرخ یادگیری، انجام بهینه‌سازی محلی یا جلوگیری از گیر کردن در نقاط مینیمم موجود، اهمیت بیشتری به انتقال سریع‌تر یا پایدار تر به مینیمم بدهند.

در مورد انتخاب الگوریتم بهینه‌سازی، تجربه، آزمون و خطا، و خصوصیات مدل و مساله می‌تواند به تصمیم نهایی کمک کند. بهینه‌سازی مناسب می‌تواند منجر به آموزش بهتر و سریع‌تر مدل شود، اما همیشه لازم است نکاتی همچون افزایش یا کاهش نرخ یادگیری را نیز در نظر گرفت.

برای آموزش مدل MLP بر روی دیتاست ، می‌توان از بهینه‌سازی‌های مختلفی مانند RMSprop یا SGD نیز استفاده کرد. انتخاب بهینه‌سازی مناسب بستگی به ویژگی‌های خاص مدل، مساله و دیتاست دارد.

- RMSPROP یک روش بهینه‌سازی است که از شبکه عصبی برای آموزش با داده‌های بزرگ استفاده می‌شود. این الگوریتم از نسبت تغییرات گرادیان را برای هر وزن استفاده می‌کند تا مقدار learning rate را تطبیق دهد. با استفاده از این متد، روشی موثر برای جلوگیری از شلیک زودرس هنگام یادگیری عمیق است. ولی طبق نتایج و مباحث یاد گرفته ادام از ان بهتر است زیرا یک روش ترکیبی از این و یک الگوریتم دیگر است پس قطعا این مورد از لیست انتخاب ها حذف می شود.

- SGD_M یک نوع از روش Stochastic Gradient Descent (SGD) است که از مفهوم momentum برای سرعت بخشیدن به فرایند یادگیری استفاده می‌کند. این الگوریتم از مفهوم گذشته گرادیان‌ها برای بهبود سرعت یادگیری استفاده می‌کند تا از مشکلات سرعت کوهیدن گرادیان در بهینه‌سازی SGD معمولی کاسته شود. این روش می‌تواند بهبود قابل توجهی در سرعت و کیفیت یادگیری شبکه‌های عصبی داشته باشد. پس این روش نیز از SGD بهتر است اما باز هم از ادام ضعیف تر زیرا طبق تئوریات ادام ترکیبی از انهاست و بهتر عمل می کند. حتی با جایگذاری انها به جای ادام از دقت بالای ۹۰ درصدی کاسته شد!

- Adam یک الگوریتم بهینه‌سازی است که ترکیبی از روش‌های RMSprop و Momentum است. Adam با استفاده از میانگین ریاضی و تجهیز شده‌ی گرادیان‌ها بهبودی بهینه‌سازی بخصوص در مسائل با مقیاس حداقلی دارد.

همانطور که در درس ماشین لرنینگ خواندیم این الگوریتم به طور کلی می‌تواند به صورت موثری در مقایسه با الگوریتم‌های دیگر عمل کند و به سرعت و کارآیی مدل کمک کند.

نوع توابع هزینه و معیارها (Metrics) بستگی به نوع مسأله یادگیری ماشین دارد.

"لاس (loss)" میزان خطای تخمینی مدل در هر مرحله از آموزش است که به منظور بهبود عملکرد مدل کاهش داده می‌شود.

برخی از توابع هزینه معروف شامل:

۱. Binary Crossentropy (دسته‌بندی دودویی)

- استفاده معمولی برای مسائل تصمیم‌گیری دودویی است.

۲. Categorical Crossentropy (دسته‌بندی چند دسته‌ای)

- معمولا برای آموزش مدل‌هایی که باید داده‌ها را به یکی از چند دسته تقسیم کنند، مورد استفاده قرار می‌گیرد.

۳. Mean Squared Error (خطا میانگین مربعات)

- معمولا برای مسائل رگرسیون استفاده می‌شود.

۴. Kullback-Leibler Divergence (انحراف کولباک-لایبلر)

- برای مدل‌های توزیع احتمالاتی و یادگیری نظارت شده به‌خصوص مسائل تولید محتوا مانند مولدهای مقابله‌ای (GANs) استفاده می‌شود.

"متریک (metrics)" به معنای معیارهایی است که برای ارزیابی عملکرد مدل در هنگام آموزش یا آزمون استفاده می‌شود، مانند دقت، دقت خاصیتی و ...

معیارها نیز برای ارزیابی مدل استفاده می‌شوند. علاوه بر دقت، معیارهای دیگری نیز وجود دارند که می‌توان در مورد عملکرد مدل استفاده کرد مانند:

- فراخوانی (recall)

- دقت (precision)

- اف اسکور (F1-score)

- ماتریس درهم‌ریختگی (confusion matrix) و...

با توجه به نوع مسأله و نوع داده‌ها، انتخاب صحیح توابع هزینه و معیارهای مناسب بسیار حیاتی است. اما انتخاب categorical_crossentropy و accuracy در این مسئله ی خاص مناسب ترین حالت می تواند باشد.

- loss=categorical_crossentropy: این بخش مشخص می‌کند که برای اندازه‌گیری خطا یا هزینه در حین آموزش از تابع هزینه‌ی "categorical_crossentropy" استفاده شود. این تابع مخصوص کاربردی است که برای مسائل دسته‌بندی چند دسته‌ای مناسب است.

البته که یک نکته دیگر هم داریم:

اگر داده‌ها به صورت integer labels هستند، انتخاب sparse_categorical_crossentropy مناسب است. اما اگر داده‌ها تبدیل به one-hot encode شده‌اند، انتخاب categorical_crossentropy صحیح است.

- metrics=['accuracy']: این بخش به مدل مشخص می‌کند که در هر مرحله از آموزش، دقت (accuracy) را به عنوان معیار برای ارزیابی عملکرد مدل استفاده

کند. که دقت نشان دهنده درصد داده‌هایی است که به درستی تشخیص داده شده‌اند.

٦-پیش پردازش:

```
def find_bounding_rectangles_of_contours(contours):
    """Determines the bounding rectangles of the contours of the cropped
letters."""
    letter_bounding_rectangles = []
    for contour in contours:
        (x, y, w, h) = cv2.boundingRect(contour)
        if w / h > 1.25:
            half_width = int(w / 2)
            letter_bounding_rectangles.append((x, y, half_width, h))
            letter_bounding_rectangles.append((x + half_width, y, half_width, h))
        else:
            letter_bounding_rectangles.append((x, y, w, h))
    return letter_bounding_rectangles


def preprocess_CAPTCHA(img):
    """Takes a CAPTCHA image and thresholds it."""
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray_with_border = cv2.copyMakeBorder(gray, 8, 8, 8, 8, cv2.BORDER_REPLICATE)
    preprocessed = cv2.threshold(
        gray_with_border, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU
    )[1]
    return gray_with_border, preprocessed


def get_CAPTCHA_label(path_to_file):
    """Get the CAPTCHA text from the file name."""
    filename = os.path.basename(path_to_file)
    label = filename.split(".")[0]
    return label


def CAPTCHA_to_gray_scale_and_bounding_rectangles(captcha_image_file):
    """Take a CAPTCHA and output a grayscale version as well as the bounding
rectangles of its cropped letters."""
    image = cv2.imread(captcha_image_file)
    gray, preprocessed = preprocess_CAPTCHA(image)
    contours = cv2.findContours(
        preprocessed.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE
```

```
    )
    contours = contours[0]
    letter_bounding_rectangles = find_bounding_rectangles_of_contours(contours)
    letter_bounding_rectangles = sorted(letter_bounding_rectangles, key=lambda x:
x[0])
    return gray, letter_bounding_rectangles
```

۱. preprocess_CAPTCHA(img): این تابع یک تصویر CAPTCHA را گرفته و آن را به تصویر سیاه و سفید تبدیل کرده و سپس با استفاده از روش thresholding آن را پیش‌پردازش می‌کند.

۲. get_CAPTCHA_label(path_to_file) (label): این تابع برچسب CAPTCHA را از نام فایل استخراج می‌کند.

۳. find_bounding_rectangles_of_contours(contours): این تابع مستطیل‌های محدود کننده اطراف حاشیه‌های (contours) حروف کاندید را تعیین می‌کند.

۴. CAPTCHA_to_gray_scale_and_bounding_rectangles(captcha_image_file): این تابع یک تصویر CAPTCHA را گرفته، آن را به تصویر خاکستری تبدیل می‌کند و مستطیل‌های محدود کننده حاشیه‌های حروف را محاسبه می‌کند.

یعنی این توابع برای پردازش تصاویر کپچا، تبدیل آن‌ها به تصاویر خاکستری و استخراج مستطیل‌های حاشیه‌های حروف کپچا برای استفاده در مراحل بعدی مانند تشخیص کاراکترها و حل CAPTCHA مورد استفاده قرار می‌گیرند.

۷-خروجی نهایی:

```
        gray, letter_bounding_rectangles = CAPTCHA_to_gray_scale_and_bounding_rectangles(
            CAPTCHA
        )
        predictions = []
[81]


        for letter_bounding_rectangle in letter_bounding_rectangles:
            x, y, w, h = letter_bounding_rectangle
            letter_image = gray[y - 2 : y + h + 2, x - 2 : x + w + 2]
            letter_image = resize_image_to_dimensions(letter_image, 20, 20)
            letter_image = np.expand_dims(letter_image, axis=2)
            letter_image = np.expand_dims(letter_image, axis=0)
            prediction = NN_model.predict(letter_image)
            letter = label_binarizer.inverse_transform(prediction)[0]
            predictions.append(letter)
[82]

...   1/1 [==============================] - 0s 345ms/step
      1/1 [==============================] - 0s 46ms/step
      1/1 [==============================] - 0s 45ms/step


        predicted_captcha_text = "".join(predictions)
        print("Predicted CAPTCHA text is: {}".format(predicted_captcha_text))
        print("CAPTCHA text is: {}".format(CAPTCHA.split("\\")[-1].split(".")[0]))
[83]

...   Predicted CAPTCHA text is: wyw
      CAPTCHA text is: 2b827


    Wrong ans for khat khati captcha! to make it better ==> see our captcha Proj
```

که تشخیص کپچا کاملا غلط صورت می گیرد و دیگر در مدل های غیر ساده این کدها و الگوریتم و روال اموزش جوابگو نبوده و باید سعی در بهبود و حتی تغییر آن باشیم که در پروژه‌ی درس به طور کامل به تغییرات تمرین پرداختیم...!

٨- مراجع

[1] https://github.com
[2] https://stackoverflow.com/questions
[3] https://www.wikipedia.org/
[4] Research paper on CAPTCHA breaking techniques
[5] Python libraries for image processing and machine learning.
[6] https://colab.research.google.com/
[7] https://www.tensorflow.org/guide/
[8] https://pandas.pydata.org/
[9] https://keras.io/
[10] https://github.com/topics/captcha-recognition
https://medium.com/@manvi./captcha-recognition-using-convolutional-neural-network-d191ef91330e