به نام خدا

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر

یادگیری عمیق با کاربرد در بینایی ماشین و پردازش صوت

تمرین شماره یک

علی الهی
۸۱۰۶۹۶۳۳۶

بهار ۱۴۰۰

فهرست

چکیده

هدف این تمرین، آشنایی با شبکه های عصبی feed forward و convolutional است. در سوال اول یک شبکه عصبی فقط با استفاده از کتابخانه Numpy به منظور پردازش تصاویر برای پیش‌بینی سن، جنسیت و قومیت پیاده‌سازی شده. سوال دوم به منظور آشنایی با لایه های convolution و pooling طراحی شده و در سوال سوم به تمرین back propagation در لایه های convolutional پرداخته می‌شود.

سوال ۱

مقدمه

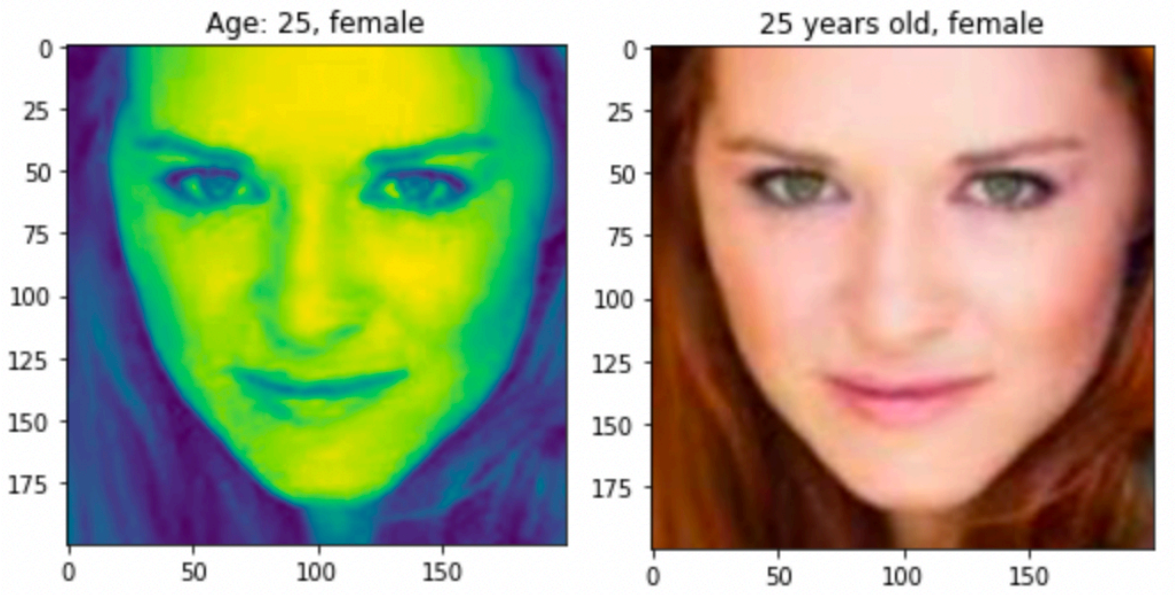
هدف این مسئله، پیش‌بینی قومیت، جنسیت و سن افراد با استفاده از تصویر صورت آن‌ها است. ابزار مورد استفاده در حل این مسئله، شبکه عصبی Fully connected است که توسط کتابخانه Numpy پیاده‌سازی شده. همچنین به بررسی پارامترهای شبکه و انواع activation function ها و loss function ها پرداخته شده و تاثیر آن‌ها بر دقت[۱] و خطا[۲] بررسی می‌شود.

معرفی مجموعه داده

در این مسئله از مجموعه داده[۳] UTKfacesبا حدود ۲۴۰۰۰ تصویر از افراد در سنین ۱ تا ۱۱۶ سال با ۵ قومیت استفاده شده. برای آموزش شبکه از ۸۲٪ و از باقی داده ها به عنوان مجموعه تست[۴] استفاده شده است.

پیش پردازش

تصاویر با استفاده از کتابخانه python image library (PIL) خوانده شده و با استفاده از numpy array ذخیره شدند. ابتدا تصاویر از RGB به gray scale تبدیل شدند (از ابعاد ۲۰۰*۲۰۰*۳ به ۲۰۰*۲۰۰) سپس به منظور دستیابی به scaling مناسب، تمامی مقادیر پیکسل‌ها به ۲۵۵ تقسیم شدند. در نهایت به منظور کاهش ابعاد تصاویر، از الگوریتم PCA استفاده شد. (از ۴۰۰۰۰ به ۱۲۸) لازم به ذکراست که برای پیاده‌سازی این الگوریتم از کتابخانه sci-kit lear استفاده شد.



شکل۰- راست: پیش از پیش‌پردازش، چپ: پس از **grayscale** و **normalization**

۱ loss
۲ accuracy
۳ dataset
۴ test set

معرفی شبکه عصبی پیاده‌سازی شده

کد شبکه عصبی پیاده‌سازی شده و نتایج بدست آمده در این Git-hub موجود است.

در فایل MyNetwork.ipynb شبکه عصبی به همراه activation function ها (از جمله sigmoid، leaky-ReLU) و loss function ها (از جمله L2 و softmax + cross-entropy) به منظور حل مسائل regression و classification پیاده‌سازی شدند. همچنین این شبکه به گونه ای طراحی شده است که پارامتر های شبکه از جمله تعداد نورون‌های هر لایه، تعداد لایه ها، توابع فعال‌سازی برای هر لایه، تابع خطا، moment، learning rate و initial weight قابل تنظیم هستند. همچنین تابع get_network_info اطلاعات مربوط به شبکه را نمایش می‌دهد.

وزن ها

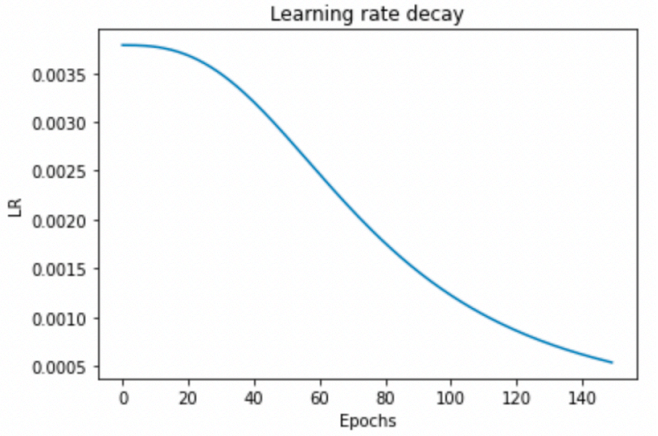
در شبکه پیاده‌سازی شده به چهار روش می‌توان وزن‌ها و بایاس ها را initialize کرد.

- با صفر
- با اعداد رندم بسیار کوچک (حدود $۱۰^{-۳}$)
- با اعداد رندم کوچک (حدود $۱۰^{-۲}$)
- با اعداد رندم نسبتا بزرگ (حدود $۱۰^{-۱}$)

تاثیر initialization با وزن‌های مختلف در ادامه بررسی خواهد شد.

ضریب یادگیری[۱]

ضریب یادگیری را می‌توان به دو صورت ثابت و متغیر تعریف کرد. در تابع set_training_param، میتوان مقدار ضریب یادگیری را تعیین کرد یا از حالت "AUTO" استفاده کرد که در آن با افزایش تعداد epoch، ضریب یادگیری کم می‌شود. برای پیاده‌سازی Learning rate decay از روشی مشابه time-based decay استفاده شده که رابطه آن به صورت $lr = {}^{a}/_{epoch^{n} + b}$ است و کاهش آن با افزایش تعداد epoch ها در نمودار زیر قابل مشاهده است.



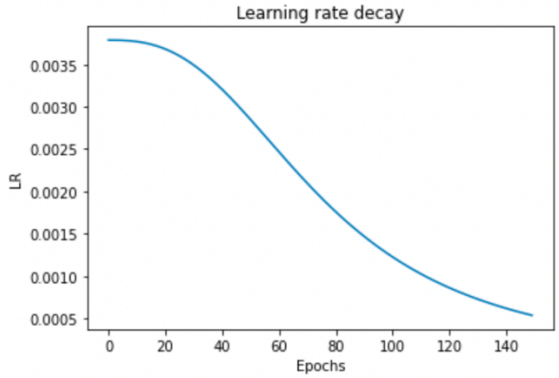
شکل۱- تغییرات ضریب یادگیری

۱ learning rate

بررسی نتایج قسمت اول: پیش‌بینی سن

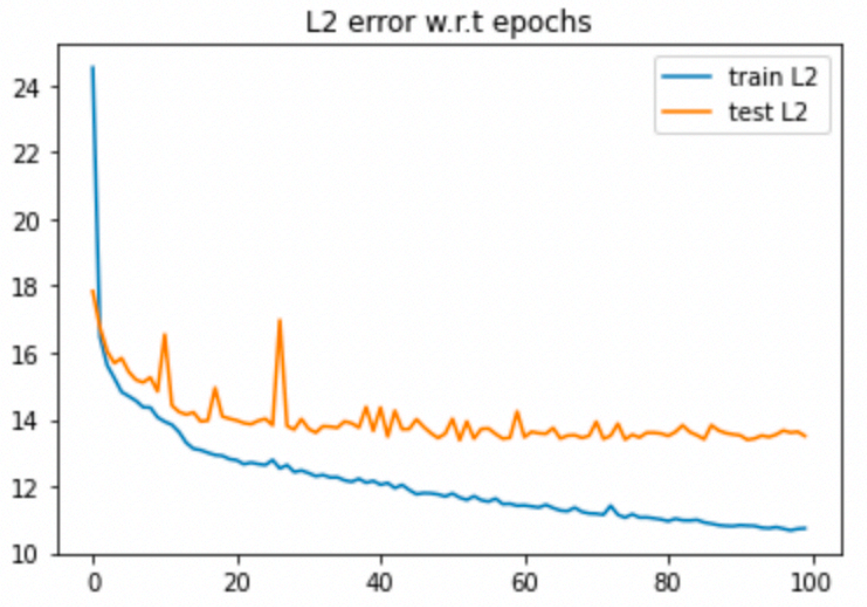
در این قسمت به آموزش یک شبکه عصبی fully connected و feed forward برای پیش‌بینی سن افراد پرداخته می‌شود. یک شبکه سه لایه به این منظور آموزش داده شد. اطلاعات مربوط به شبکه در ادامه قابل مشاهده است.

```
3 layers:
35 neurons      activation function: LeakyReLU  Weights are initialized by small random numbers.
6 neurons       activation function: LeakyReLU  Weights are initialized by small random numbers.
1 neurons       activation function: LeakyReLU  Weights are initialized by small random numbers.
Momentum:       0.99
Loss Function:   L2
```



شکل۲- اطلاعات مربوط به شبکه عصبی (پیش‌بینی سن)

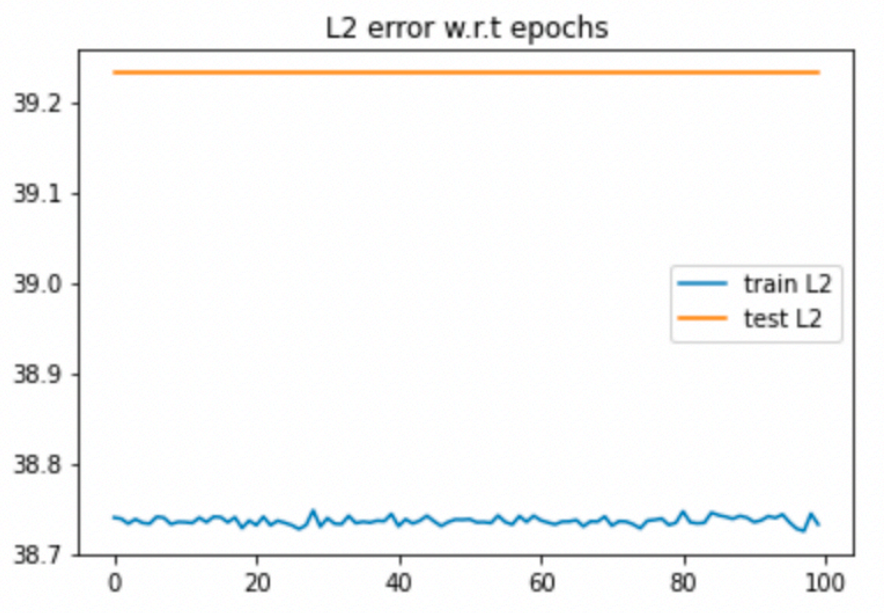
تغییرات خطا برای مجموعه داده های تست و آموزش در نمودار فوق قابل مشاهده است. قابل ذکر است که برای جلو گیری از افزایش خطای تست، از early stopping استفاده شده. همانطور که قابل ملاحظه است، RMSE برای داده های آموزش و تست به مقادیر ۱۰ و ۱۳ همگرا شده است (به این معنی‌ست که سنین با اختلاف ۱۳ سال پیش‌بینی می‌شوند.) و این در حالی است که رگرسور Logistic کتابخانه Sci-Kit learn این پیش‌بینی را با خطای ۱۵.۷۲ انجام می‌دهد.



شکل۳- خطای **RMSE** طی **epoch** ها برای پیش‌بینی سن

بررسی شبکه با وزن‌های اولیه صفر

در صورتی که مقدار اولیه صفر به وزن ها داده شود، شبکه پیشرف نخواهد کرد زیرا در back propagation مشتق خروجی خطی نورون‌ها نسبت به ورودی نورون برابر وزن است و در صورتی که وزن ها صفر باشند، backprop tensor در صفر ضرب شده و وزن‌ها آپدیت نمی‌شوند.

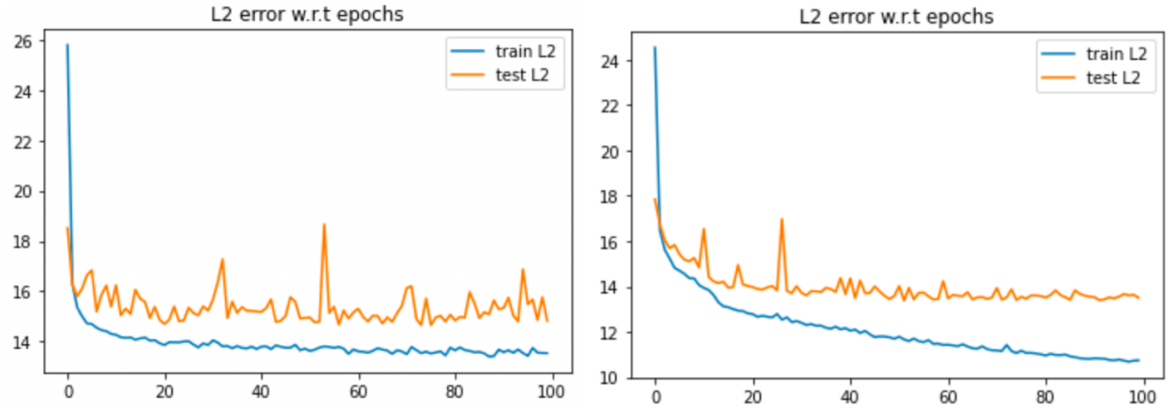


شکل۴- مقدار دهی اولیه صفر

$$\frac{d(w_1x_1 + w_2x_2 + \cdots)}{dx_1} = w_1 \overset{w_1=0}{\Longrightarrow} 0$$

بررسی ضرایب یادگیری ثابت و متغیر (در حال کاهش)

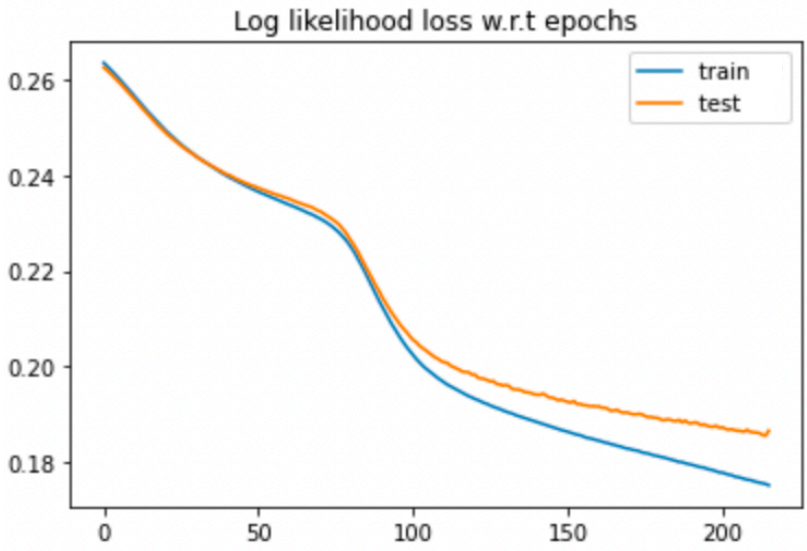
در صورتی که ضریب یادگیری طی آموزش شبکه ثابت بماند، نمودار loss برحسب epoch بسیار noisy خواهد بود و دلیل آن این است که با آموزش بیشتر شبکه، باید قدم ها کوچک باشند تا به راحتی به نقطه بهینه همگرا شود و حول آن نوسان نکند. دو شبکه یکی به ضریب یادگیری ثابت و دیگری با ضریب یادگیری متغیر (شکل ۱) روی این مجموه داده ها آموزش داده شدند و نمودار تغییرات loss نسبت به epoch ها برای آن‌ها در ادامه قابل مشاهده است. سایر پارامترهای این دو شبکه یکسان هستند.



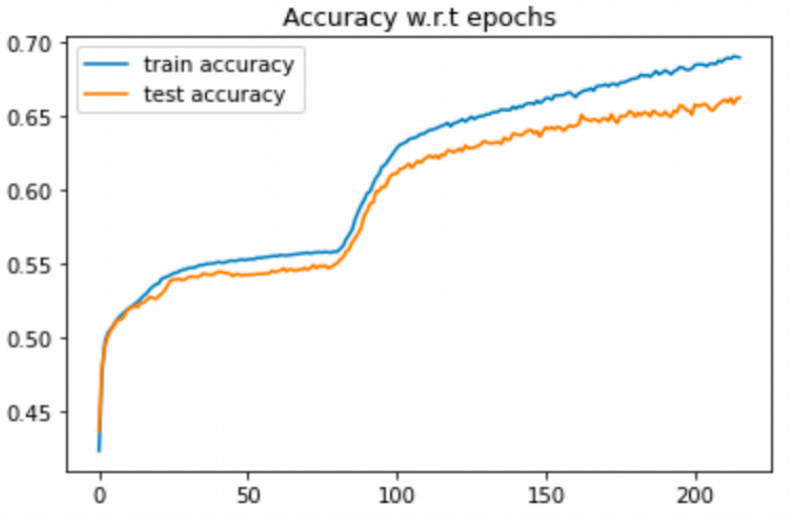
شکل۴- راست: ضریب یادگیری متغیر. چپ: ضریب یادگیری ثابت

بررسی نتایج قسمت دوم: پیش‌بینی قومیت

این بار از شبکه عصبی برای حل مسئله classification استفاده شده است. در لایه خروجی از تابع softmax برای محاسبه احتمال هر کلاس و از negative log-likelihood (یا cross-entropy) به عنوان تابع خطا استفاده شده. همچنین از تابع فعال‌ساز leaky-ReLU در تمامی لایه‌ها بجز لایه آخر و ضریب یادگیری ثابت و برابر ۰.۰۰۲ استفاده شده. در ادامه نمودار های accuracy w.r.t epoch و loss w.r.t epoch قابل مشاهده است. قابل ملاحظه است که دقت برای داده های آموزش و تست به مقادیر ۶۸ و ۶۵ همگرا شده است.



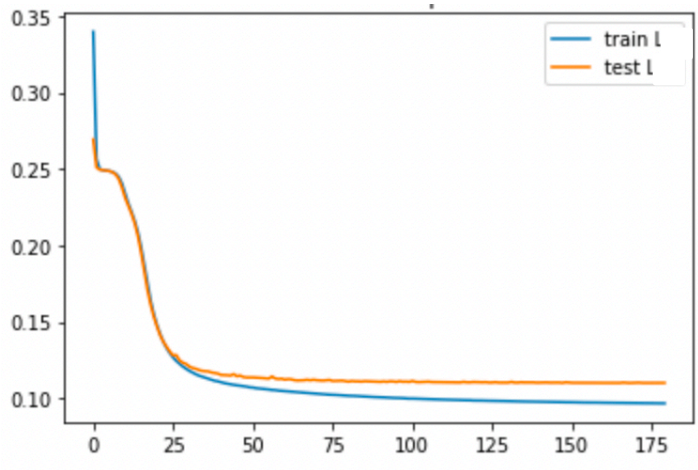
شکل۵- خطا طی **epoch** ها برای پیش‌بینی قومیت



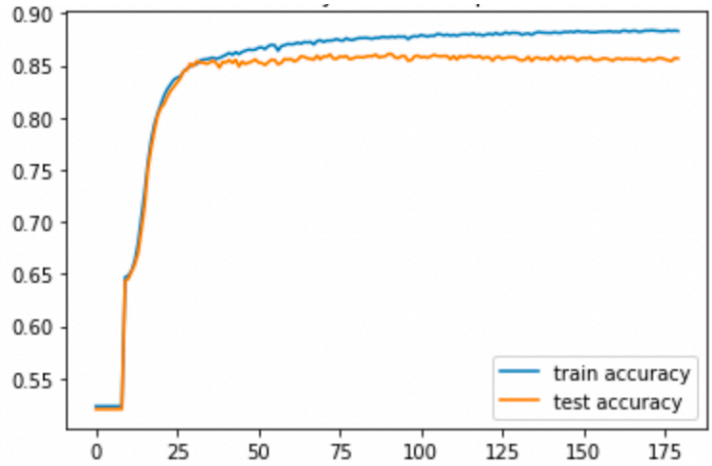
شکل۶- دقت طی **epoch** ها برای پیش‌بینی قومیت

بررسی نتایج قسمت سوم: پیش‌بینی جنسیت

حال از می‌خواهیم از شبکه عصبی برای پیش‌بینی جنسیت استفاده کنیم. در این مسئله از خطای MSE استفاده شده و ضریب یادگیری نیز نرخ کاهشی دارد. همچنین از تابع فعال‌ساز leaky-ReLU استفاده شده. قابل ملاحظه است که دقت برای داده های آموزش و تست به مقادیر ۸۸ و ۸۶ همگرا شده است.



شکل۷- خطا طی **epoch** ها برای پیش‌بینی جنسیت



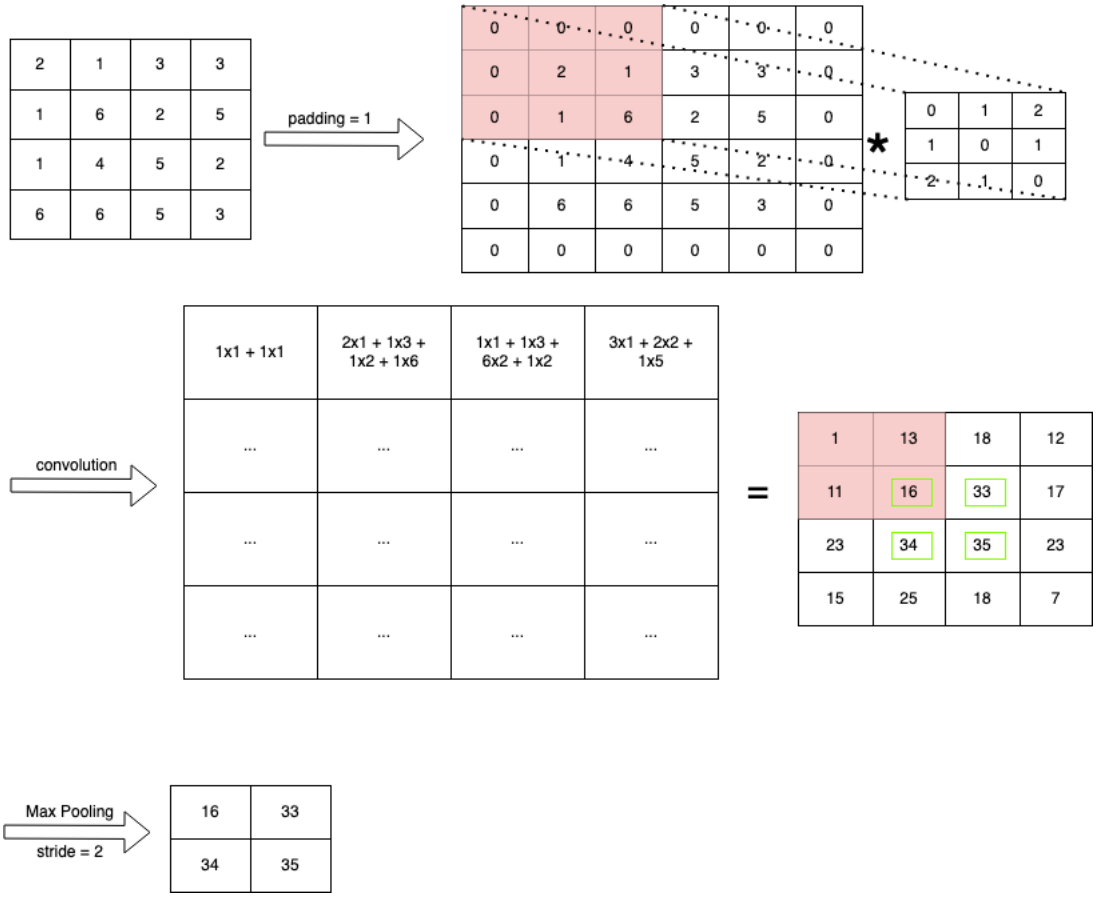
شکل۸- دقت طی **epoch** ها برای پیش‌بینی جنسیت

سوال ۲

برای حل این سوال در فلدر codes، در فایل Q#2 – Convolution.ipynb توابعی برای لایه‌های convolutional و pooling پیاده‌سازی شده است. برای این سوال، ابتدا تصویر با padding = 1 در فیلترهای ۳*۳ convolve می‌شود و پس از آن 2*2 max pooling با stride = 2 انجام می‌شود. این فرآیند برای هر سه فیلتر تکرار می‌شود.

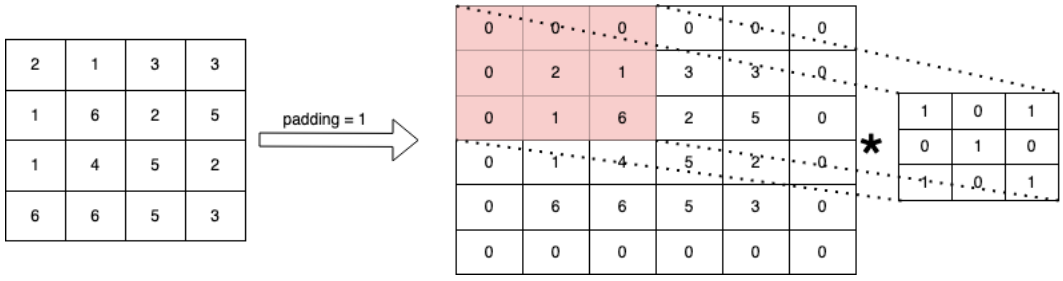
$$\text{Input image } (4*4*1) \xrightarrow{*(3*3*3)\ with\ padding=1} (4*4*3) \xrightarrow{\max pooling\ with\ stride\ =\ 2} (2*2*3)$$

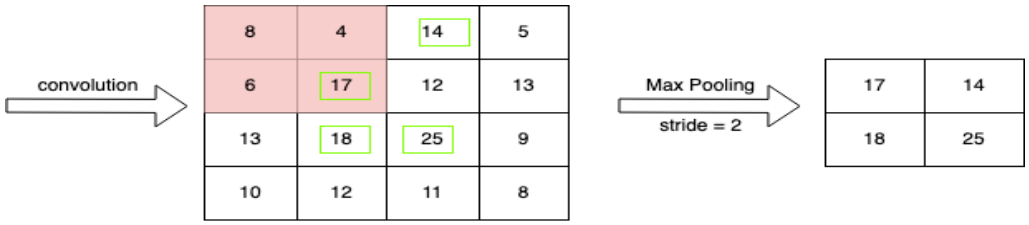
اولین فیلتر:



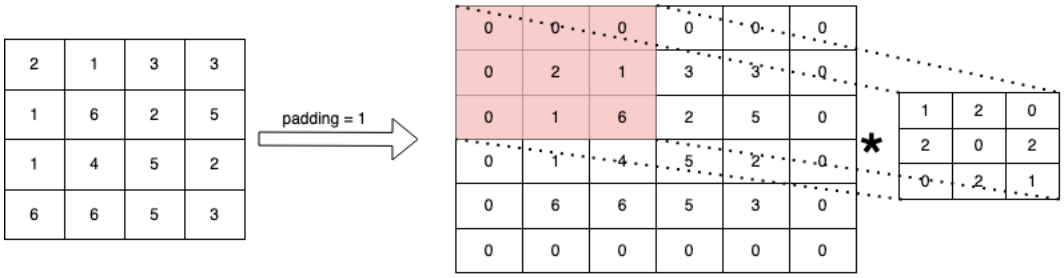
شکل ۱۰- انجام محاسبات **convolution** و **pooling** فیلتر اول

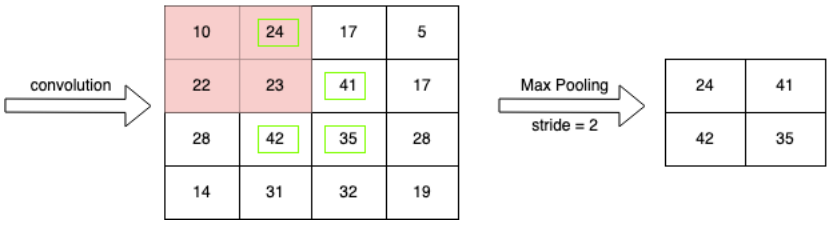
توجه داشته باشید که ابتدا محاسبات توسط کد python صورت گرفته است سپس این تصاویر توسط ابزار diagrams.net ساخته شده‌اند. در ادامه محاسبات مربوط به دو فیلتر دیگر قابل مشاهده است.





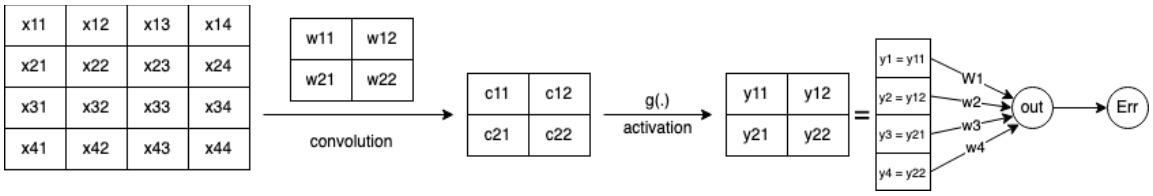
شکل ۱۱- انجام محاسبات **convolution** و **pooling** فیلتر دوم





شکل ۱۲- انجام محاسبات **convolution** و **pooling** فیلتر سوم

سوال ۳



شکل۱۳- شبکه سوال ۳

در این سوال قصد داریم محاسبات back propagation را برای شبکه فوق انخام داده و مقادیر $derr/_{dw}$ را برای تمامی وزن های شبکه محاسبه کنیم.

$$\frac{d(error)}{d(out)} = \frac{d(\frac{1}{2}(t-out)^2)}{d(out)} = t - out; \quad \frac{d(out)}{dw_i} = \frac{d(\sum w_i y_i)}{dw_i} = y_i$$

با دانستن موارد فوق برای وزن های لایه آخر خواهیم داشت:

$$\frac{d(error)}{dw_1} = \frac{d(error)}{d(out)}\frac{d(out)}{dw_1} = (t-out)y_1$$

$$\frac{d(error)}{dw_2} = \frac{d(error)}{d(out)}\frac{d(out)}{dw_2} = (t-out)y_2$$

$$\frac{d(error)}{dw_3} = \frac{d(error)}{d(out)}\frac{d(out)}{dw_3} = (t-out)y_3$$

$$\frac{d(error)}{dw_4} = \frac{d(error)}{d(out)}\frac{d(out)}{dw_4} = (t-out)y_4$$

برای لایه convolutional داریم:

$$\frac{dy_{ij}}{dc_{ij}} = g'(c_{ij}); \quad \frac{d(out)}{dy_i} = \frac{d(\sum w_i y_i)}{dy_i} = w_i; \quad \frac{dc_{ij}}{dw_{kl}} = \frac{d(\sum c_{ij} w_{kl})}{dw_{kl}} = x_{ab}$$

$$\begin{aligned}\frac{d(error)}{dw_{11}} = \frac{d(error)}{d(out)}\Bigg(&\frac{d(out)}{dy_1(=y_{11})}\frac{dy_{11}}{dc_{11}}\frac{dc_{11}}{dw_{11}} + \frac{d(out)}{dy_2(=y_{12})}\frac{dy_{12}}{dc_{12}}\frac{dc_{12}}{dw_{11}} \\ &+ \frac{d(out)}{dy_3(=y_{21})}\frac{dy_{21}}{dc_{21}}\frac{dc_{21}}{dw_{11}} + \frac{d(out)}{dy_4(=y_{22})}\frac{dy_{22}}{dc_{22}}\frac{dc_{22}}{dw_{11}}\Bigg) \\ &= (t-out)(w_1 g'(c_{11})x_{11} + w_2 g'(c_{12})x_{13} + w_3 g'(c_{21})x_{31} + w_4 g'(c_{22})x_{33})\end{aligned}$$

$$\begin{aligned}\frac{d(error)}{dw_{12}} = \frac{d(error)}{d(out)}\Bigg(&\frac{d(out)}{dy_1(=y_{11})}\frac{dy_{11}}{dc_{11}}\frac{dc_{11}}{dw_{12}} + \frac{d(out)}{dy_2(=y_{12})}\frac{dy_{12}}{dc_{12}}\frac{dc_{12}}{dw_{12}} \\ &+ \frac{d(out)}{dy_3(=y_{21})}\frac{dy_{21}}{dc_{21}}\frac{dc_{21}}{dw_{12}} + \frac{d(out)}{dy_4(=y_{22})}\frac{dy_{22}}{dc_{22}}\frac{dc_{22}}{dw_{12}}\Bigg) \\ &= (t-out)(w_1 g'(c_{11})x_{12} + w_2 g'(c_{12})x_{14} + w_3 g'(c_{21})x_{32} + w_4 g'(c_{22})x_{34})\end{aligned}$$

$$\frac{d(error)}{dw_{11}} = \frac{d(error)}{d(out)}\left(\frac{d(out)}{dy_1(=y_{11})}\frac{dy_{11}}{dc_{11}}\frac{dc_{11}}{dw_{21}} + \frac{d(out)}{dy_2(=y_{12})}\frac{dy_{12}}{dc_{12}}\frac{dc_{12}}{dw_{21}} + \frac{d(out)}{dy_3(=y_{21})}\frac{dy_{21}}{dc_{21}}\frac{dc_{21}}{dw_{21}} + \frac{d(out)}{dy_4(=y_{22})}\frac{dy_{22}}{dc_{22}}\frac{dc_{22}}{dw_{21}}\right)$$
$$= (t - out)(w_1 g'(c_{11})x_{21} + w_2 g'(c_{12})x_{23} + w_3 g'(c_{21})x_{41} + w_4 g'(c_{22})x_{43})$$

$$\frac{d(error)}{dw_{12}} = \frac{d(error)}{d(out)}\left(\frac{d(out)}{dy_1(=y_{11})}\frac{dy_{11}}{dc_{11}}\frac{dc_{11}}{dw_{22}} + \frac{d(out)}{dy_2(=y_{12})}\frac{dy_{12}}{dc_{12}}\frac{dc_{12}}{dw_{22}} + \frac{d(out)}{dy_3(=y_{21})}\frac{dy_{21}}{dc_{21}}\frac{dc_{21}}{dw_{22}} + \frac{d(out)}{dy_4(=y_{22})}\frac{dy_{22}}{dc_{22}}\frac{dc_{22}}{dw_{22}}\right)$$
$$= (t - out)(w_1 g'(c_{11})x_{22} + w_2 g'(c_{12})x_{24} + w_3 g'(c_{21})x_{42} + w_4 g'(c_{22})x_{44})$$

پیوست ۱: روند اجرای برنامه

تمامی کد ها در فلدر codes قرار دارند و به ضمیمه گزارش آپلود شده اند. همچنین در git hub قابل ملاحظه می‌باشند. شبکه عصبی مربوط به سوال اول در فایل MyNetwork.ipynb و حل سوال اول در فایل دیگری با نام UTKFaces regression and classification.ipynb قرار دارند. برای اجرای برنامه باید دیتاست در فلدر codes قرار داده شود. همچنین کد مربوط به سوال دوم نیز در فایل Q#2 – Convolution.ipynb قرار دارد.

مراجع

- 2D Convolution using Python & NumPy