

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

<u>3</u>	<u>چکیده</u>
<u>4</u>	<u>سوال ۱</u>
4	مقدمه
4	معرفی مجموعه داده
4	پیش پردازش
5	معرفی شبکه عصبی پیاده سازی شده
5	وزن ها
5	ضریب یادگیری
6	بررسی نتایج قسمت اول: پیش بینی سن
7	بررسی شبکه با وزن های اولیه صفر
7	بررسی ضرایب یادگیری ثابت و متغیر (در حال کاهش)
8	بررسی نتایج قسمت دوم: پیش بینی قومیت
8	بررسی نتایج قسمت سوم: پیش بینی جنسیت
<u>10</u>	<u>سوال ۲</u>
<u>12</u>	<u>سوال ۳</u>
<u>14</u>	<u>پیوست ۱: روند اجرای برنامه</u>
<u>15</u>	<u>مراجع</u>

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 پیش بینی سن، جنسیت و قومیت پیاده سازی 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 می شود.

مقدمه

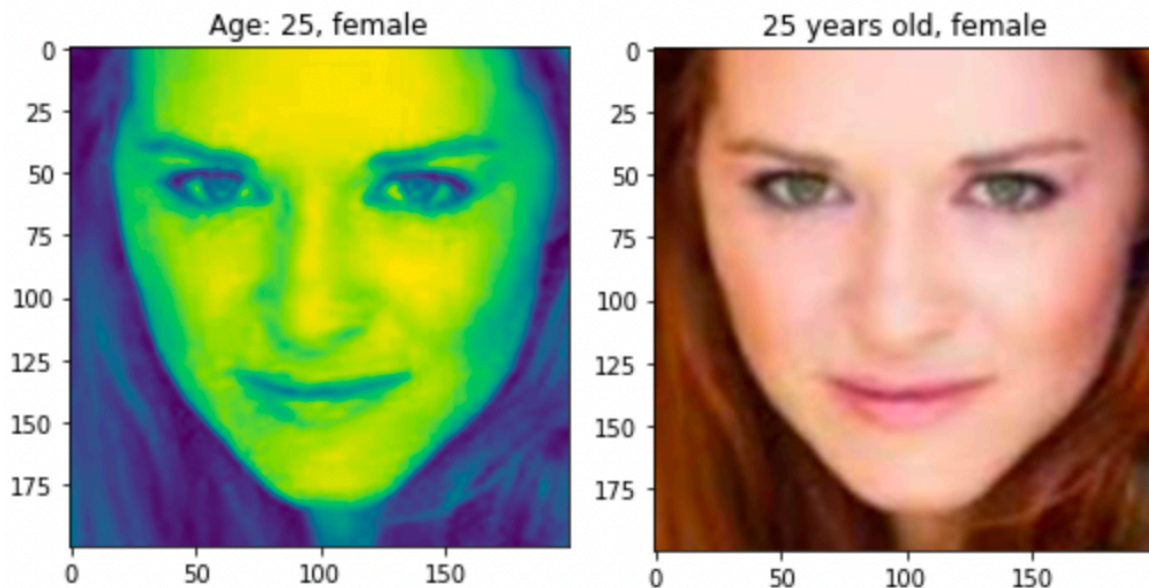
هدف این مسئله، پیش‌بینی قومیت، جنسیت و سن افراد با استفاده از تصویر صورت آن‌ها است. ابزار مورد استفاده در حل این مسئله، شبکه عصبی Fully connected است که توسط کتابخانه Numpy پیاده‌سازی شده. همچنین به بررسی پارامترهای شبکه و انواع activation function ها و loss function ها پرداخته شده و تاثیر آن‌ها بر دقت^۱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

معرفی مجموعه داده

در این مسئله از مجموعه داده^۳ UTKfaces با حدود ۲۴۰۰۰ تصویر از افراد در سنین ۱ تا ۱۱۶ سال با ۵ قومیت استفاده شده. برای آموزش شبکه از ۸۲٪ و از باقی داده‌ها 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 $3 \times 200 \times 200$ به 200×200) 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 40000 به 128) لازم به ذکر است که برای پیاده‌سازی این الگوریتم از کتابخانه sci-kit lear استفاده شد.



شکل ۰ - 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 پارامترهای 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.

وزن‌ها

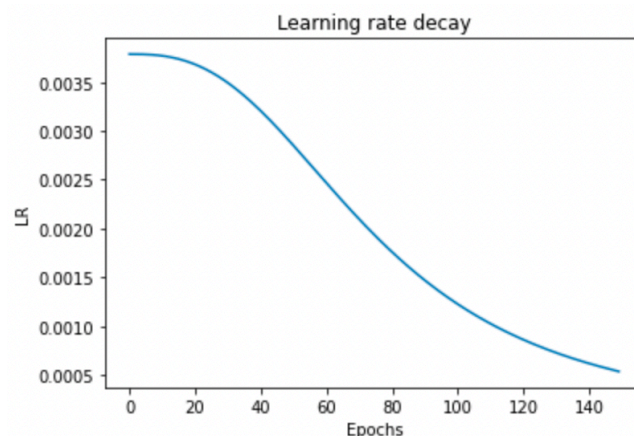
در شبکه پیاده‌سازی شده به چهار روش می‌توان وزن‌ها و بایاس‌ها را initialize کرد.

- با صفر
- با اعداد رندم بسیار کوچک (حدود 10^{-3})
- با اعداد رندم کوچک (حدود 10^{-2})
- با اعداد رندم نسبتاً بزرگ (حدود 10^{-1})

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.

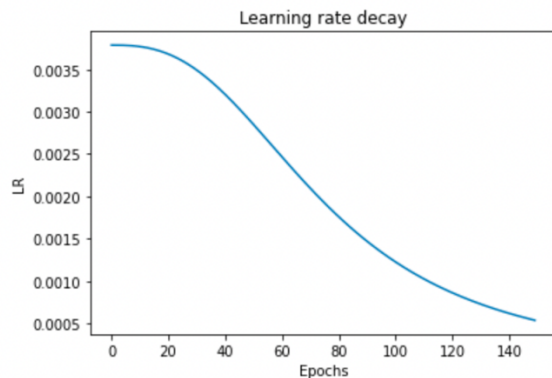


شکل ۱- تغییرات ضریب یادگیری

بررسی نتایج قسمت اول: پیش‌بینی سن

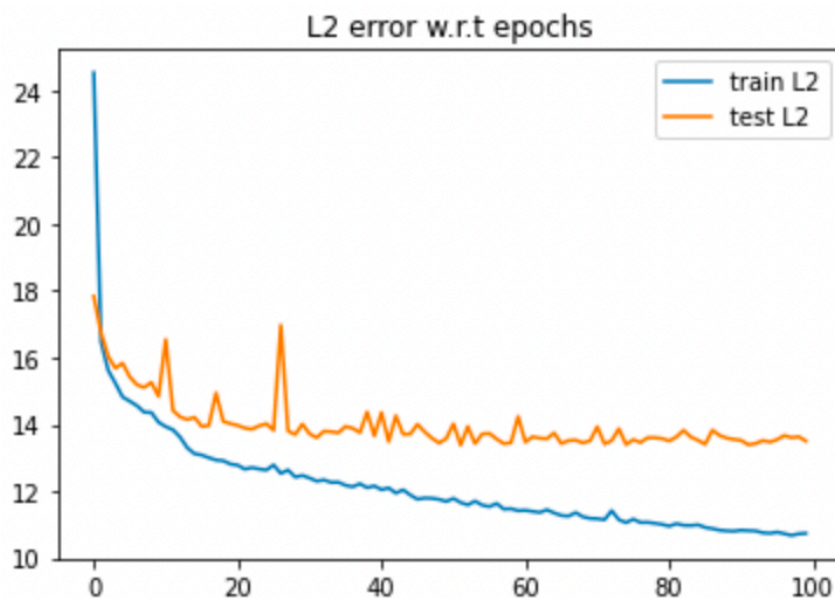
در این قسمت به آموزش یک شبکه عصبی fully connected و feed forward برای پیش‌بینی سن افراد پرداخته می‌شود. یک شبکه سه لایه به این منظور آموزش داده شد. اطلاعات مربوط به شبکه در ادامه قابل مشاهده است.

```
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
```



شکل ۲- اطلاعات مربوط به شبکه عصبی (پیش‌بینی سن)

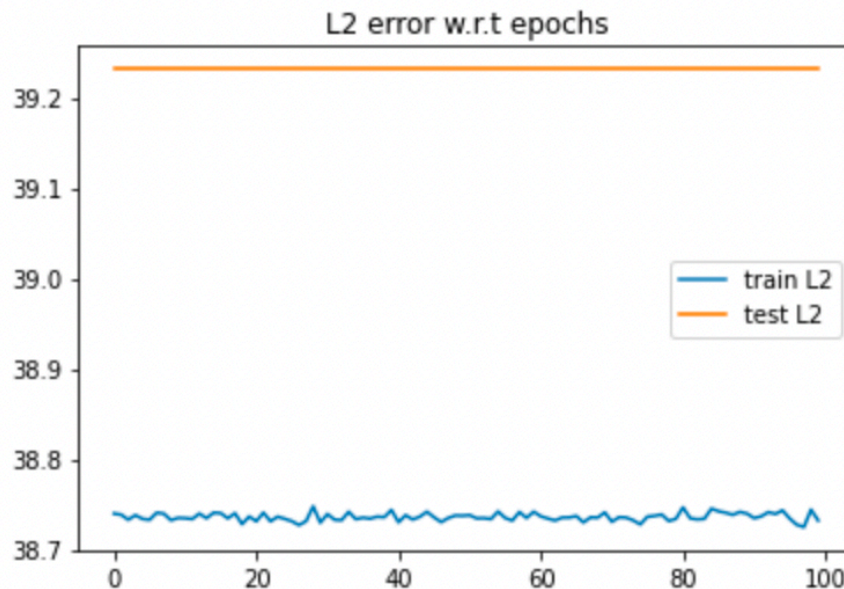
تغییرات خطا برای مجموعه داده های تست و آموزش در نمودار فوق قابل مشاهده است. قابل ذکر است که برای جلوگیری از افزایش خطای تست، از early stopping استفاده شده. همانطور که قابل ملاحظه است، RMSE برای داده های آموزش و تست به مقادیر ۱۰ و ۱۳ همگرا شده است (به این معنی است که سنین با اختلاف ۱۳ سال پیش‌بینی می‌شوند)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



شکل ۳- خطای RMSE طی epoch ها برای پیش‌بینی سن

بررسی شبکه با وزن‌های اولیه صفر

در صورتی که مقدار اولیه صفر به وزن‌ها 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 وزن‌ها صفر باشند، backprop tensor در صفر ضرب شده و وزن‌ها آپدیت نمی‌شوند.

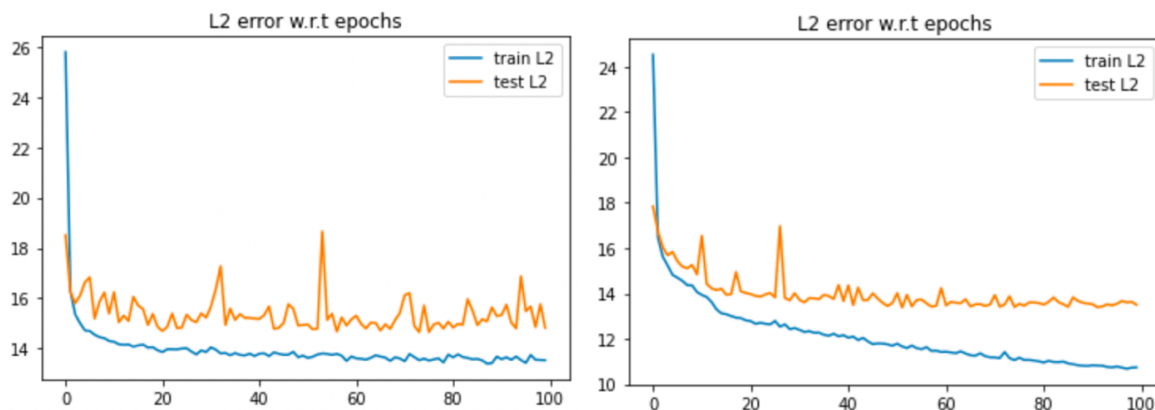


شکل ۴- مقدار دهی اولیه صفر

$$\frac{d(w_1x_1 + w_2x_2 + \dots)}{dx_1} = w_1 \xrightarrow{w_1=0} 0$$

بررسی ضرایب یادگیری ثابت و متغیر (در حال کاهش)

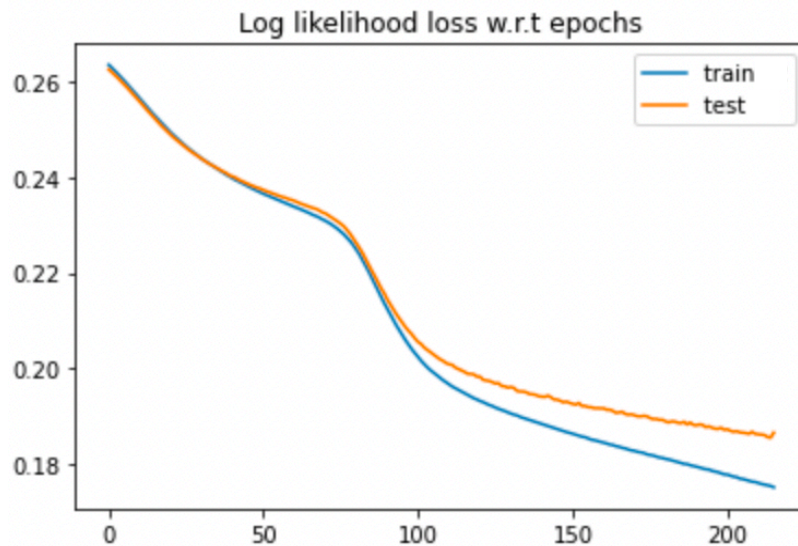
در صورتی که ضریب یادگیری طی آموزش شبکه ثابت بماند، نمودار loss برحسب epoch بسیار noisy خواهد بود و دلیل آن این است که با آموزش بیشتر شبکه، باید قدم‌ها کوچک باشند تا به راحتی به نقطه بهینه همگرا شود و حول آن نوسان نکند. دو شبکه یکی به ضریب یادگیری ثابت و دیگری با ضریب یادگیری متغیر (شکل ۱) روی این مجموعه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



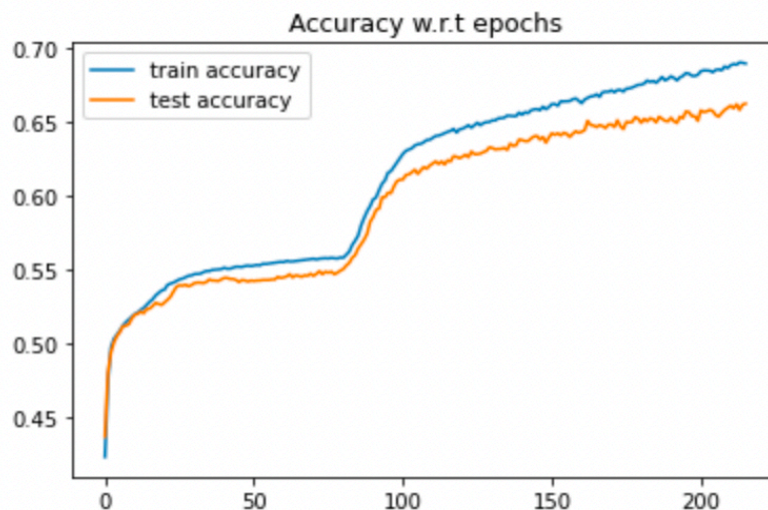
شکل ۴- راست: ضریب یادگیری متغیر. چپ: ضریب یادگیری ثابت

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 ضرب یادگیری ثابت و برابر ۰.۰۰۲ استفاده شده. در ادامه نمودارهای accuracy w.r.t epoch و loss w.r.t epoch قابل مشاهده است. قابل ملاحظه است که دقت برای داده‌های آموزش و تست به مقادیر ۶۸ و ۶۵ همگرا شده است.



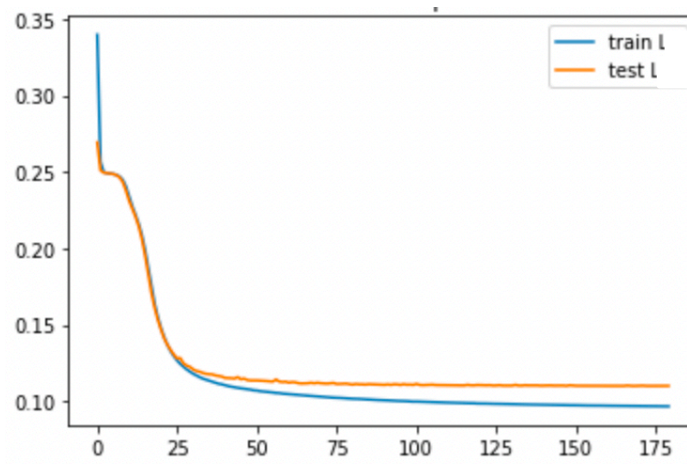
شکل ۵- خطای epoch ها برای پیش‌بینی قومیت



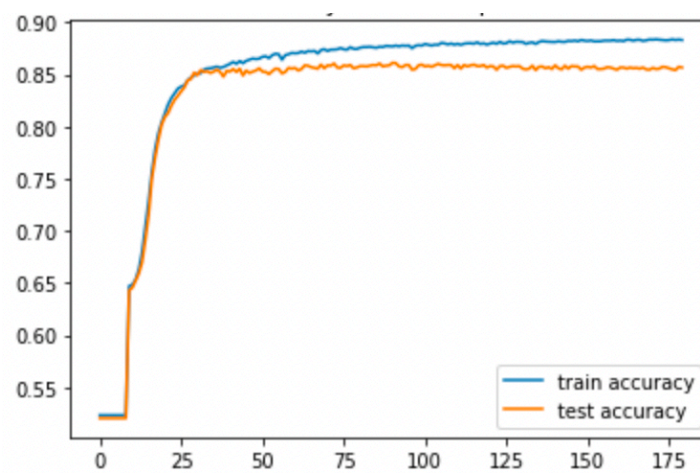
شکل ۶- دقت epoch ها برای پیش‌بینی قومیت

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 ضرب یادگیری نیز نرخ کاهشی دارد. همچنین از تابع فعال‌ساز leaky-ReLU استفاده شده. قابل ملاحظه است که دقت برای داده‌های آموزش و تست به مقادیر ۸۸ و ۸۶ همگرا شده است.



شکل ۷- خطا طی epoch ها برای پیش‌بینی جنسیت

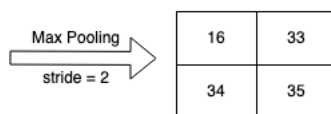
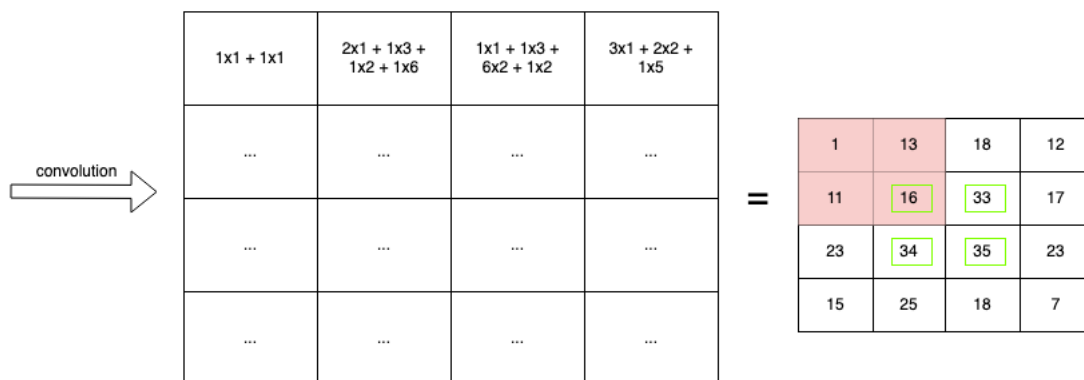
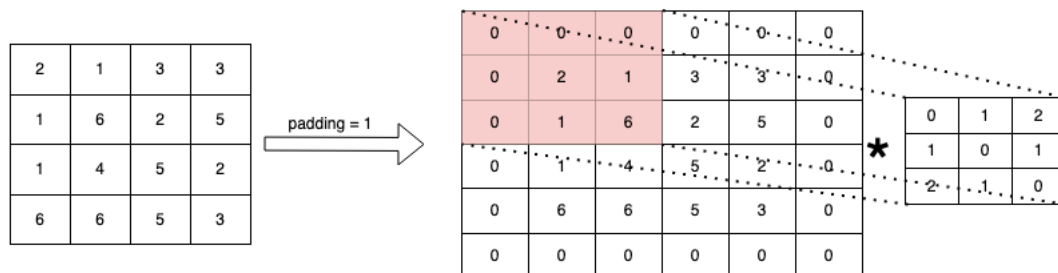


شکل ۸- دقت طی epoch ها برای پیش‌بینی جنسیت

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 $\text{padding} = 1$ در فیلترهای 3×3 convolve می‌شود و پس از آن 2×2 max pooling انجام می‌شود. این فرآیند برای هر سه فیلتر تکرار می‌شود.

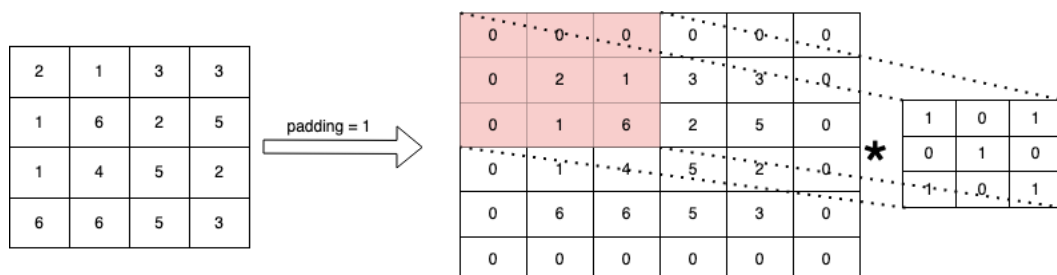
$$\text{Input image } (4 \times 4 \times 1) \xrightarrow{*(3 \times 3 \times 3) \text{ with padding}=1} (4 \times 4 \times 3) \xrightarrow{\text{max pooling with stride}=2} (2 \times 2 \times 3)$$

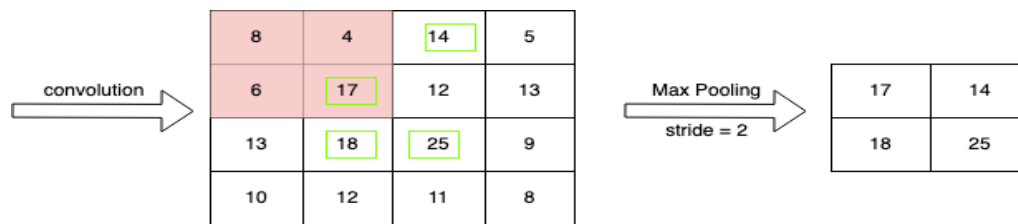
اولین فیلتر:



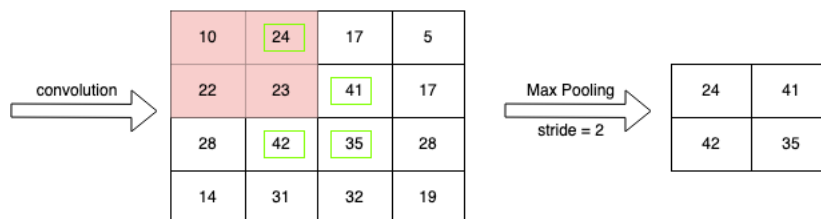
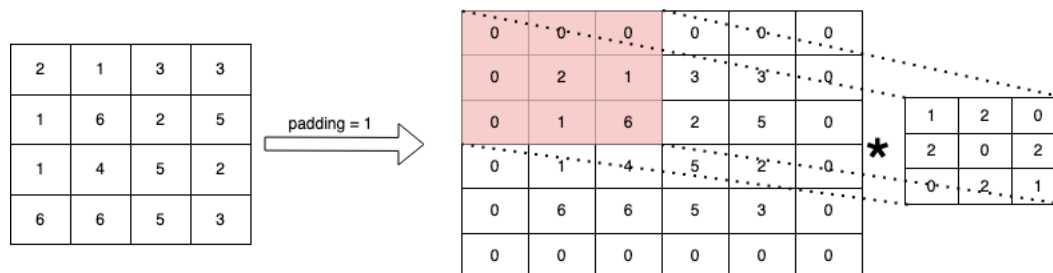
شکل ۱۰- انجام محاسبات convolution و pooling فیلتر اول

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.

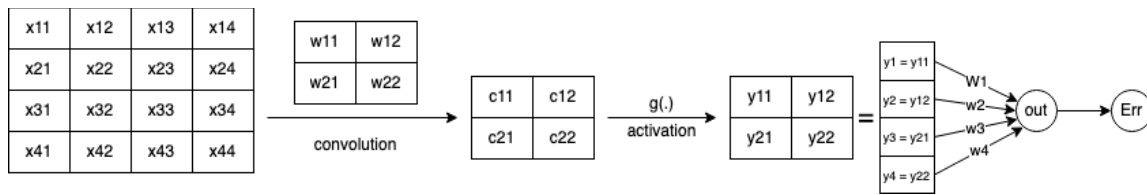




شکل ۱۱- انجام محاسبات convolution و pooling فیلتر دوم



شکل ۱۲- انجام محاسبات convolution و pooling فیلتر سوم



شکل ۱۳- شبکه سوال ۳

در این سوال قصد داریم محاسبات back propagation را برای شبکه فوق انجام 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:

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

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

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

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

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

پیوست ۱: روند اجرای برنامه

تمامی کد ها در فلدر codes قرار دارند و به ضمیمه گزارش آپلود شده اند. همچنین در [git hub](https://github.com) قابل ملاحظه می باشند. 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.

- [2D Convolution using Python & NumPy](#)