به نام خدا

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر

درس یادگیری عمیق و کاربردها

تمرین شماره ۲

امیر محمد کریمی

۸۱۰۱۹۴۳۸۳

فروردین ماه ۱۳۹۸

فهرست

بخش اول سوال ۱ ..... ۳

بخش اول سوال ۲ ..... ۴

بخش اول سوال ۳ ..... ۵

بخش اول سوال ۴ ..... ۸

بخش اول سوال ۵ ..... ۱۱

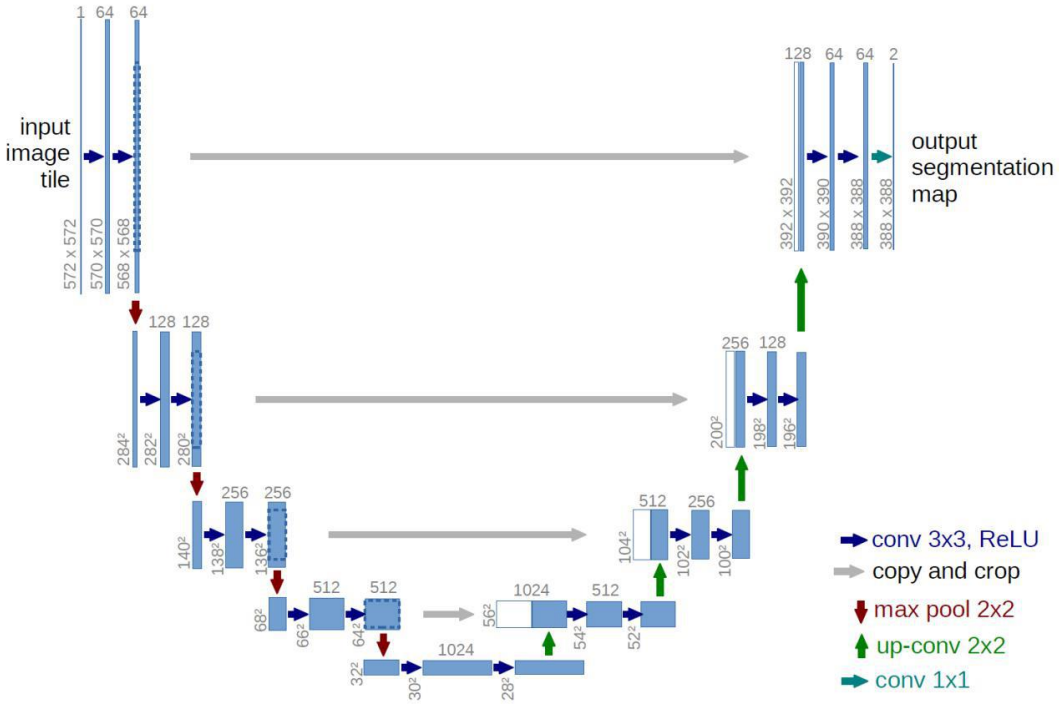
بخش دوم سوال 1 ..... ۱۳

پیوست ۲۳

مراجع ۲۴

بخش اول - سوال ۱

در این قسمت، در ابتدا شبکه‌ی U-Net را مطابق با مقاله‌ی مورد نظر به کمک ابزار pytorch پیاده‌سازی می‌کنیم. این شبکه به منظور انجام Semantic Segmentation استفاده می‌شود. ساختار این شبکه به صورت زیر می‌باشد:



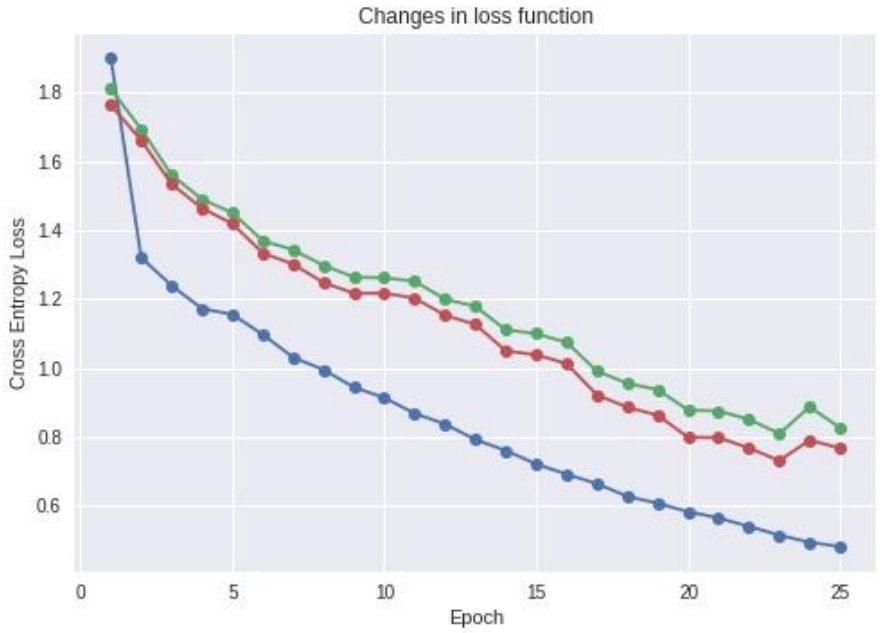
شکل ۱. ساختار شبکه‌ی U-Net

دیتاست مورد استفاده در این تمرین، CamVid می‌باشد که حاوی ۷۰۱ عکس با ابعاد ۷۲۰*۹۶۰ است. در این تمرین، ۷۲ درصد از داده‌ها را برای آموزش، ۸ درصد را برای ارزیابی (validation) و ۲۰ درصد باقی‌مانده را برای تست در نظر گرفته‌ایم.

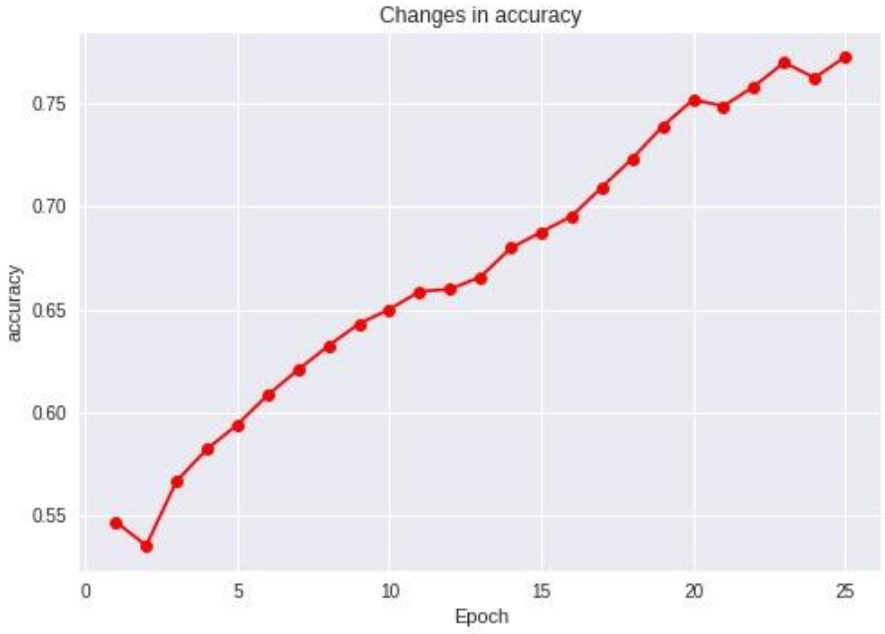
پس از طراحی شبکه‌ی عصبی مورد نظر، ابتدا برچسب داده‌های ورودی را به فرمت مورد نظر در می‌آوریم (به جای هر پیکسل، شماره‌ی کلاس آن را جایگزین می‌کنیم). سپس داده‌های آموزش را به شبکه می‌دهیم تا آموزش ببیند و در هر epoch، داده‌های تست و ارزیابی را نیز به شبکه می‌دهیم و دقت و خطای آن‌ها را نیز بررسی می‌کنیم.

لازم به ذکر است که در این تمرین، ماکزیمم تعداد epoch ممکن برای آموزش شبکه ۲۵ بوده است زیرا با توجه به عدم دسترسی بنده به سخت افزار مناسب و همچنین محدودیت google colab (که پس از ۱۲ ساعت، ماشین مجازی در اختیار قرار داده‌شده را از اختیار ما خارج می‌کند)، امکان آموزش شبکه با تعداد epoch بیشتر فراهم نبوده است.

در این سوال، شبکه‌ی U-Net را به روش Adam آموزش می‌دهیم. میزان خطا و همچنین دقت شبکه به صورت زیر است:

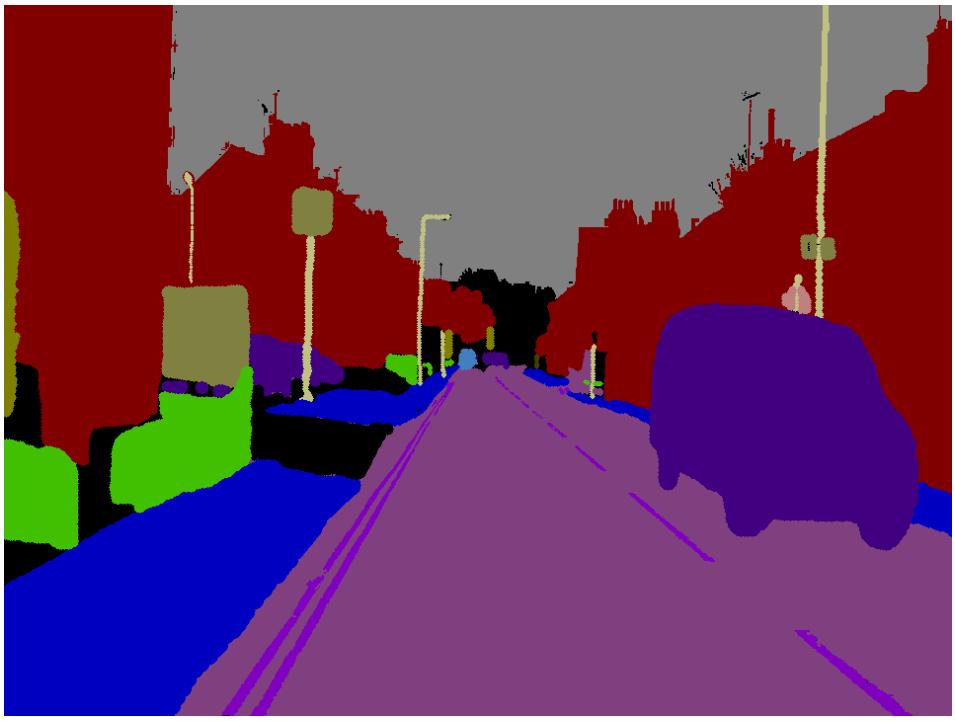


شکل ۲ . میزان خطای شبکه‌ی U-Net در ۲۵ epoch
(خطای داده‌های آموزش، ارزیابی و تست به ترتیب با رنگ‌های آبی، سبز و قرمز آمده‌اند)

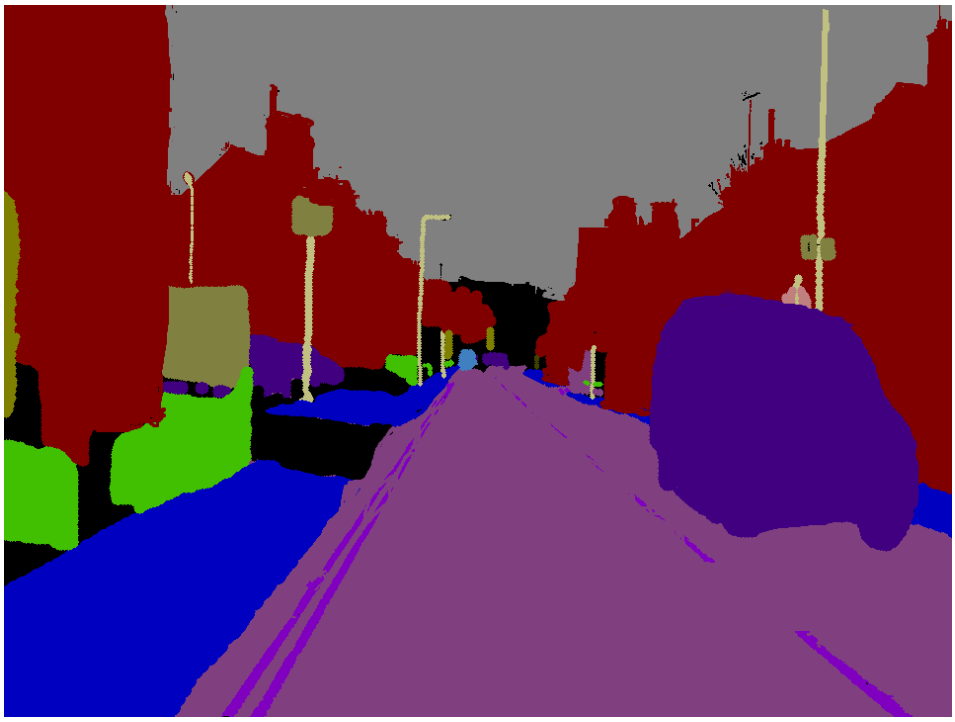


شکل ۳ . میزان دقت شبکه‌ی U-Net در ۲۵ epoch

با بررسی نمودارهای بالا مشاهده می‌کنیم که میزان دقت شبکه در نهایت به ۷۸/۸ درصد می‌رسد. سپس، یک عکس ورودی را به شبکه می‌دهیم و خروجی آن را مشاهده می‌کنیم. خروجی شبکه و خروجی مورد نظر در شکل‌های زیر نمایش داده شده است:



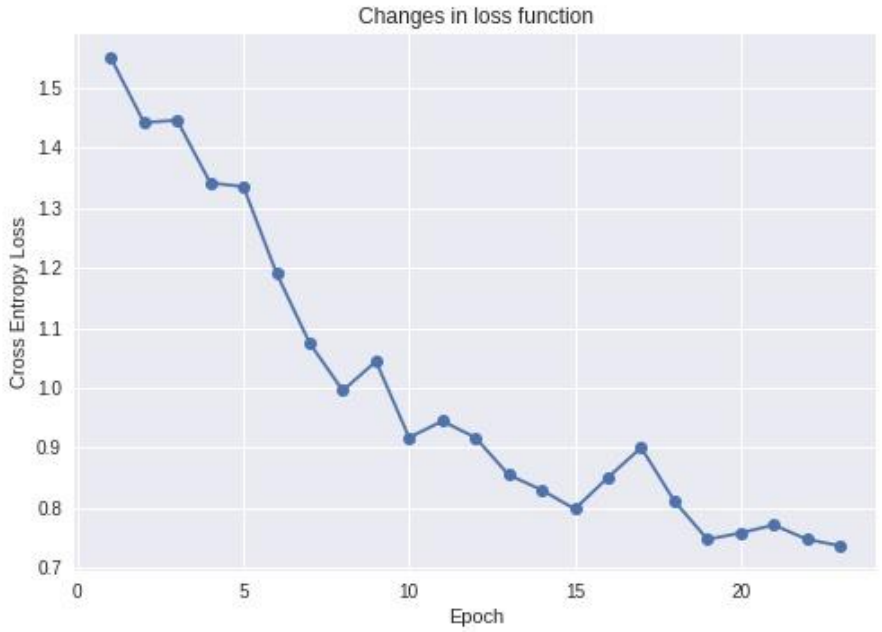
شکل ۴ . خروجی مورد نظر یک عکس تصادفی



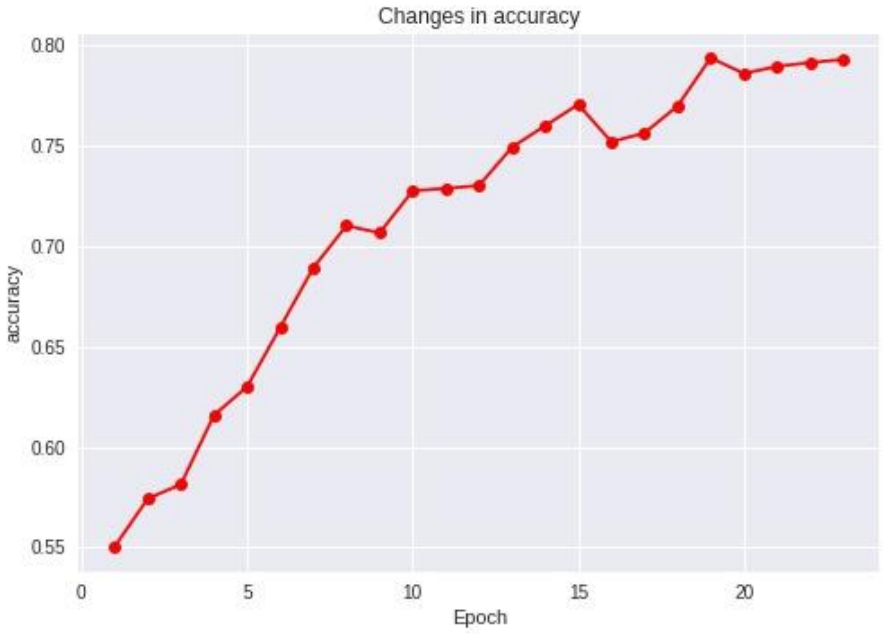
شکل ۵ . خروجی شبکه

بخش اول - سوال ۲

در این بخش، در تمامی بخش‌های شبکه، هر دو لایه کانولوشن متوالی ۳*۳ را با یک لایه کانولوشن ۵*۵ جایگزین می‌کنیم و سپس شبکه را به همان روش بخش قبل آموزش می‌دهیم. میزان خطا و دقت شبکه به صورت زیر به دست آمده است: (متاسفانه به دلیل ضعف سخت افزار که در بخش اول مفصلا توضیح داده شده است، قادر به آموزش شبکه با بیش از ۲۳ epoch نبودیم)



شکل ۶ . میزان خطای شبکه‌ی U-Net در ۲۳ epoch



شکل ۷. میزان دقت شبکه‌ی U-Net در ۲۳ epoch

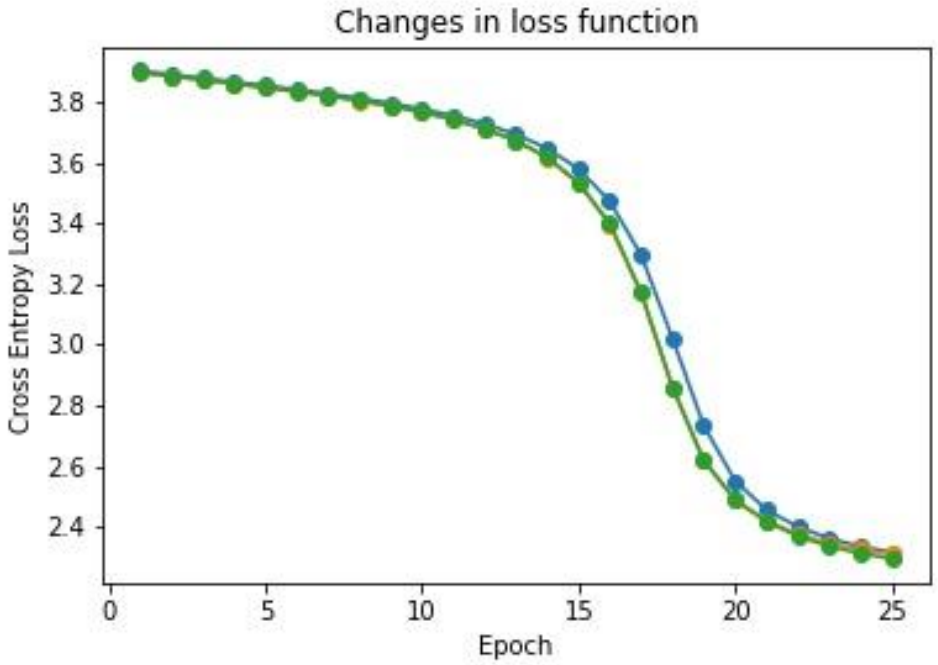
با بررسی نمودارهای بالا مشاهده می‌کنیم که دقت این شبکه در بهترین حالت به ۷۹/۷ درصد می‌رسد.

مقایسه‌ی نتایج:

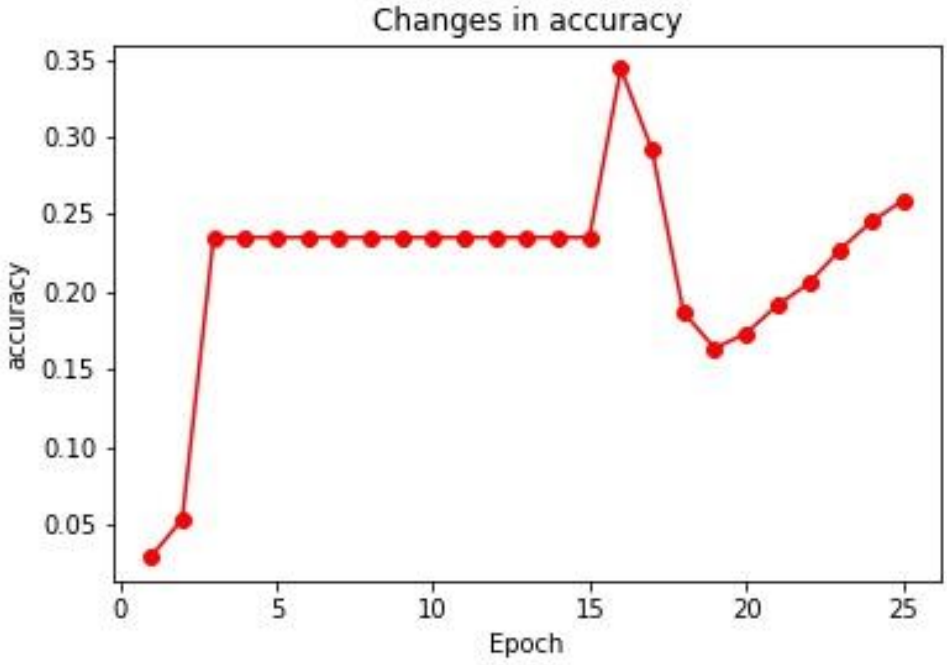
- تعداد پارامترهای شبکه‌ی دوم حدودا ۳۶ میلیون پارامتر می‌باشد در حالی که شبکه‌ی اول، حدود ۲۹ میلیون پارامتر دارد.
- میزان حافظه‌ی مورد نیاز برای شبکه‌ی دوم تقریبا نصف فضای شبکه‌ی اول می‌باشد.
- شبکه‌ی دوم زمان بیشتری را برای آموزش نیاز دارد (به دلیل تعداد پارامترهای بیشتر) و دقت آن نیز اندکی بهتر از شبکه‌ی اول می‌باشد.

بخش اول - سوال ۳

در این قسمت، روش بهینه سازی را از Adam به SGD تغییر می‌دهیم و مجددا شبکه را آموزش می‌دهیم. میزان خطا و دقت شبکه در این حالت به صورت زیر است:



شکل ۸. میزان خطای شبکه‌ی U-Net در ۲۵ epoch با روش بهینه سازی SGD (خطای داده‌های آموزش، ارزیابی و تست به ترتیب با رنگ‌های آبی، سبز و قرمز آمده‌اند)



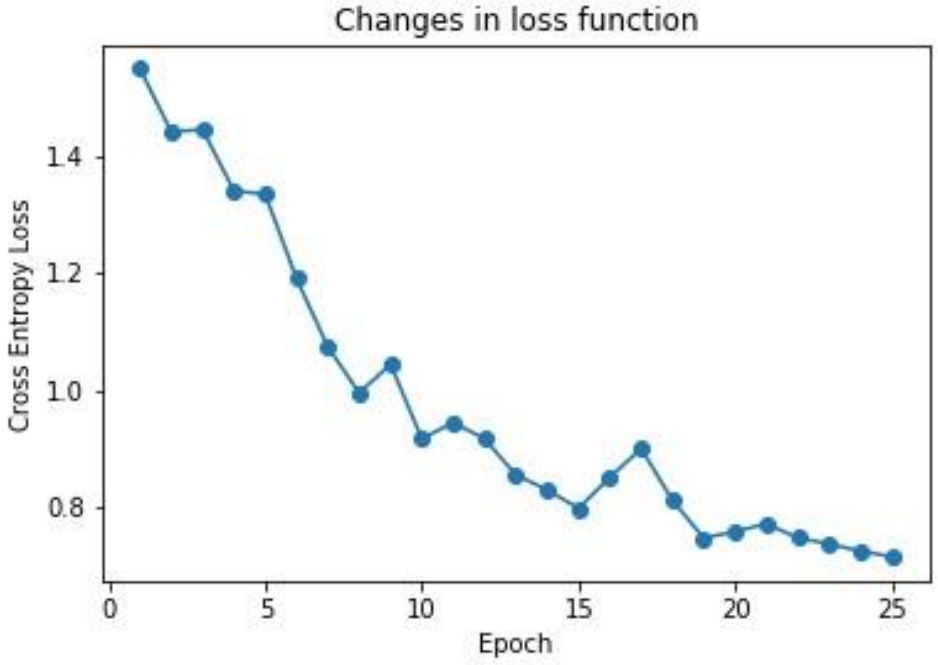
شکل ۹ . میزان دقت شبکه‌ی U-Net در ۲۵ epoch با روش بهینه سازی SGD

همانطور که مشاهده می‌شود، این روش پس از حدود ۱۵ epoch شروع به پیشرفت می‌کند و تا این زمان به دقت ۳۴ درصد می‌رسد و میزان loss آن از این epoch شروع به کاهش یافتن به صورت جدی می‌کند و سرعت همگرایی آن کمتر از روش Adam می‌باشد. البته در صورتی که در تعداد زیادی epoch قصد مقایسه این دو شبکه را داشته باشیم، روش بهینه سازی SGD بهتر عمل خواهد کرد.

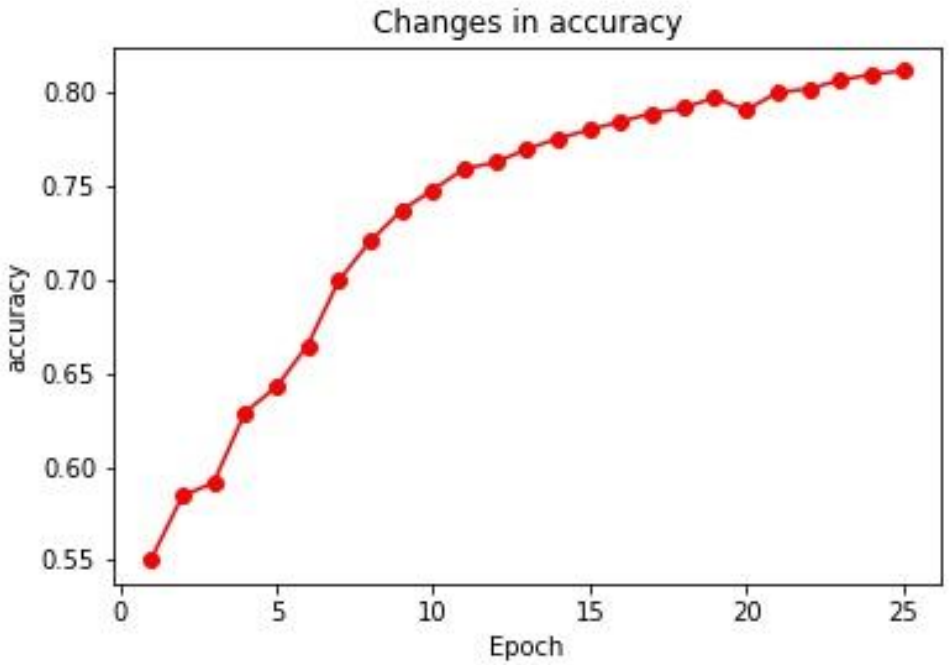
بخش اول - سوال ۴

در این قسمت، هر لایه‌ی کانولوشن و max pooling را با یک لایه‌ی کانولوشن با stride = 2 جایگزین می‌کنیم و مجددا شبکه را آموزش و بررسی می‌کنیم. میزان خطا و دقت شبکه در این حالت به صورت زیر می‌باشد:

1



شکل ۱۰. میزان خطای شبکه‌ی U-Net

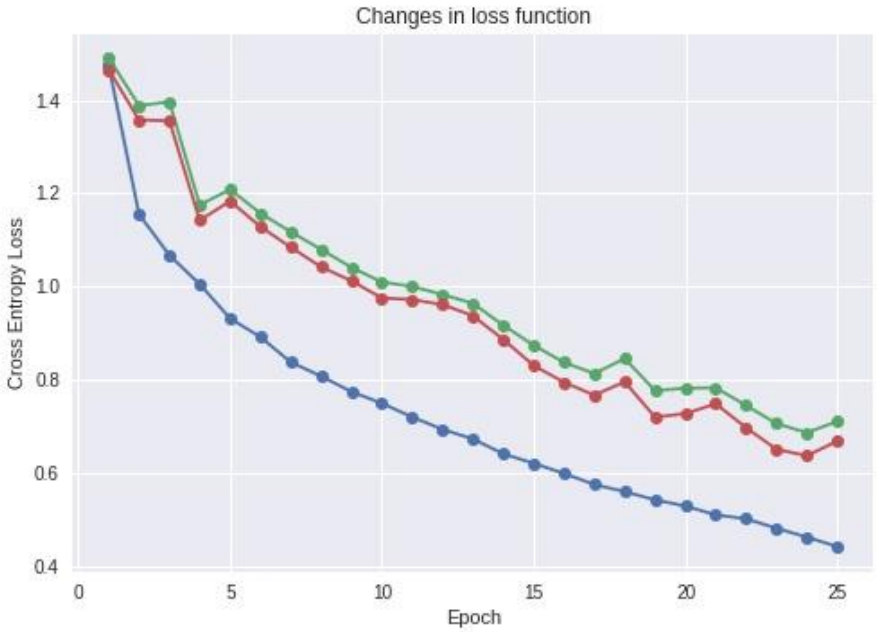


شکل ۱۱ . میزان دقت شبکه‌ی U-Net

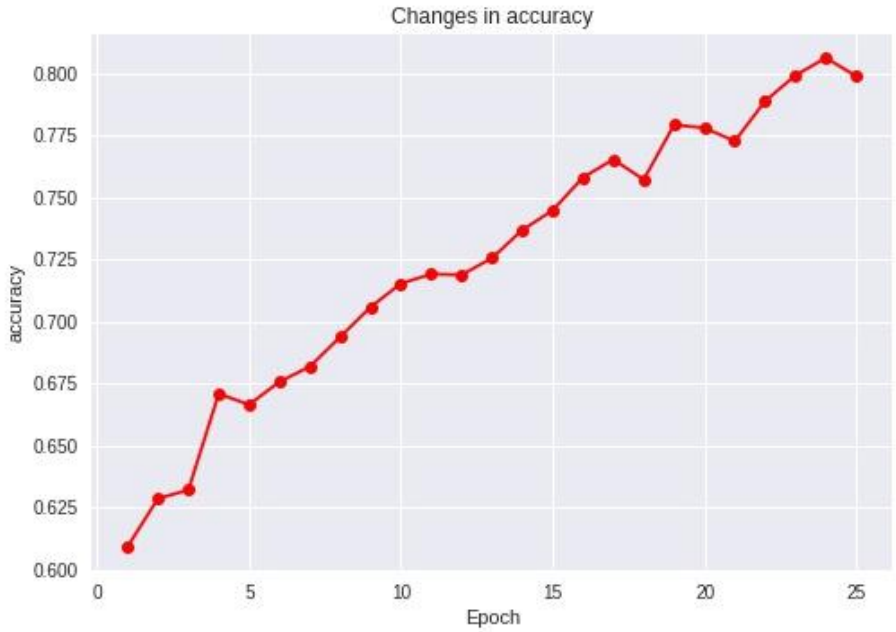
مشاهده می‌کنیم که دقت شبکه نسبت به حالت اول حدود ۴ درصد افزایش یافته است و به ۸۲ درصد رسیده است. علت آن نیز این است که عمل max pooling یک عمل ثابت است اما لایه‌ی کانولوشن با stride=2 یک لایه‌ای است که امکان آموزش دارد و در واقع به مجموعه‌ای از پارامترها که توسط داده‌های ورودی آموزش داده می‌شوند، وابسته است. به همین دلیل، امکان بهبود نتیجه‌ی شبکه با استفاده از لایه‌ی کانولوشن با stride=2 نسبت به حالتی که از max pooling که یک عمل مشخص و بدون پارامتر آموزشی می‌باشد، میسر می‌باشد.

بخش اول - سوال ۵

در این بخش، پیش از اعمال تابع ReLU در لایه‌های کانولوشنی، عمل Batch Normalization را روی داده‌ها انجام می‌دهیم. خروجی شبکه به صورت زیر است:



شکل ۱۲ . میزان خطای شبکه‌ی U-Net در ۲۵ epoch به همراه Batch Normalization
(خطای داده‌های آموزش، ارزیابی و تست به ترتیب با رنگ‌های آبی، سبز و قرمز آمده‌اند)



شکل ۱۳ . میزان دقت شبکه‌ی U-Net در ۲۵ epoch به همراه Batch Normalization

با بررسی نتایج این بخش با بخش اول مشاهده می‌کنیم که دقت شبکه اندکی بهبود یافته است و به ۸۰ درصد رسیده است و البته همگرایی شبکه نیز اندکی سریعتر انجام می‌شود.

بخش دوم

متاسفانه این بخش را به دلیل کمبود وقت (در واقع زمانبر بودن آموزش شبکه‌های بخش اول تمرین) نتوانستم انجام دهم :(

پیوست 1: روند اجرای برنامه

برای اجرای jupyter notebook بخش اول تمرین، کافیست تا دیتاست CamVid را در گوگل درایو خود آپلود کنیم و مسیر آن را در متغیر directory ذخیره کنیم. سپس می‌توانیم تمامی cell های این فایل را به ترتیب اجرا کنیم.

فایل my_unet.py نیز حاوی شبکه مورد نظر با تمامی تنظیمات مورد نیاز برای بخش‌های مختلف تمرین است که می‌توان با تغییر پارامترهای ورودی آن در فایل jupyter مذکور به آن‌ها رسید.

مراجع

[1] Goodfellow, I., Bengio, Y. and Courville, A. (n.d.). Deep learning.

[2] Ronneberger, O., Fischer, P. and Brox, T. (2019). U-Net: Convolutional Networks for Biomedical Image Segmentation. [online] arXiv.org. Available at: https://arxiv.org/abs/1505.04597 [Accessed 1 Apr. 2019].