

به نام خدا

گزارش تمرین سری دوم یادگیری عمیق

Image Colorization

استاد درس: دکتر حامد ملک

نام دانشجو: طراوت پارت

بهار 1401

فهرست مطالب

3	پاسخ سوالات تشریحی
5	گزارش پیاده‌سازی انجام شده
5	توضیح مجموعه داده
6	پیاده سازی شبکه عصبی ابتدایی
6	ماژول downconv
7	ماژول upconv
8	ماژول bottleneck
9	ساخت شبکه basemodel
10	تنظیم پارامتر های base model
10	آزمایش 1- استفاده از مقدار NF های مختلف
12	آزمایش 2- استفاده از اندازه کرnel های مختلف
14	آزمایش 3- استفاده از leraning rate های مختلف
16	نتیجه گیری
17	اضافه کردن skip-connections
17	تنظیم پارامتر های :customUnet
17	آزمایش 1- استفاده از مقدار NF های مختلف
19	آزمایش 2- استفاده از اندازه کرnel های مختلف
22	آزمایش 3- استفاده از leraning rate های مختلف
24	نتیجه گیری
25	اضافه کردن residual blocks
26	خروجی آموزش شبکه حاوی :residual blocks
27	خروجی مدل ها

پاسخ سوالات تشریحی

1. مزایای استفاده از CNN در تسک‌های پردازش تصویر چیست؟

- استخراج فیچرهای مناسب بخش مهمی از حل مسائل هوش مصنوعی است. در داده‌های جدولی، به سادگی می‌توانیم بفهمیم که کدام ستون‌ها مهم‌تر هستند و کدام ستون‌ها را باید دور بریزیم. اما استخراج ویژگی‌ها از داده‌هایی از جمله speech, video, image و ... می‌تواند سخت باشد. در قدیم از الگوریتم‌های سنتی برای استخراج فیچرهای مهم از تصاویر استفاده می‌شد. اما این مدل‌ها نیمتوانند دقت خود را در شرایط مختلف تعیین بدھند. امروزه شبکه‌ها عصبی عمیق جای الگوریتم‌های سنتی را گرفته‌اند. این شبکه‌ها یک بازنمایی از داده‌ها را یاد می‌گیرند و دیگر نیازی نیست که ویژگی‌ها را به صورت دستی از داده‌ها استخراج کنیم.
- در شبکه‌های عصبی عمیق فرآیند پیدا کردن فیچرها به صورت سلسله مراتبی اتفاق می‌افتد. هر لایه از فیچرهای اسخراج شده در لایه‌های قبل استفاده می‌کند و فیچرهای پیچیده‌تر را می‌سازد.
- در تصاویر هر پیکسل نماینده یک نورون است. اگر از شبکه‌های عصبی feedforward استفاده می‌کردیم، باید هر نورون را به تمام نورون‌ها قبلی وصل می‌کردیم که تعداد نورون‌ها خیلی زیاد می‌شد. اما در شبکه‌های عصبی CNN هر پیکسل به ناحیه کوچکی از فیچرمپ لایه قبلی اتصال دارد.

2. آنچه در فرآیند batch normalization رخ میدهد را به صورت مختصر توضیح دهید و مزایای استفاده از آن در شبکه‌های عصبی عمیق را بیان نمایید.

یکی از کارهای مهمی که در مرحله پیش‌پردازش داده‌ها اعمال می‌شود، normalization و standardization است. با انجام این مرحله ورودی اولین لایه شبکه عصبی Normalize می‌شود. اما این اعداد در وزن‌های شبکه ضرب می‌شوند وتابع فعالیت روی آنها اعمال می‌شود. بنابراین ورودی لایه‌های بعدی شبکه لزوماً در بازه $[0, 1]$ نخواهد بود. در شبکه‌های عصبی، خروجی هر لایه ورودی لایه بعدی است. فرآیند Batch normalization روی خروجی هر لایه انجام می‌شود.

فرض کنید خروجی یک لایه مخفی $\{x_1, x_2, \dots, x_m\} = B$ باشد. برای انجام batch normalization، میانگین و واریانس مقادیر مجموعه B محاسبه می‌شود و مقادیر x_i توسط روابط نشان داده شده در زیر آپدیت می‌شود:

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$$

$$x_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i = \gamma x_i + \beta$$

دلیل استفاده از پارامترهای γ و β این است که ممکن است لزوماً نخواهیم میانگین و انحراف معیار مقادیر خروجی صفر و یک باشد. این پارامترها توسط الگوریتم یادگرفته می‌شود.

3. توضیح دهید چرا residual learning از محوشگی گرادیان (vanishing gradient) جلوگیری می‌کند؟

اگر شبکه‌های عصبی خود را خیلی deep کنیم، ممکن است که این شبکه‌ها حتی به خوبی داده‌های آموزشی را یاد نگیرند. چون ممکن است مقدار گرادیان کوچکتر و کوچکتر شوند (مطابق با ضربهای متوالی در قاعده chain rule) و در نتیجه وزن‌ها تقریباً ثابت بمانند.

فرض کنید که از skip connection ها استفاده کنیم و مقادیر لایه $2+1$ را به 1 اضافه کنیم. در بدترین حالت، شبکه تابع همانی را یاد می‌گیرد و مقادیر لایه 1 به راحتی به $2+1$ منتقل می‌شوند. یادگرفتن تابع همانی بسیار برای این شبکه آسان است. پس اضافه کردن residual connections باعث بدتر شدن یادگیری نمی‌شود.

اگر لایه‌های عمیق‌تر دانش جدیدی یادگرفته باشند، از دانش آنها استفاده می‌شود. در غیر این صورت شبکه تابع همانی را یادمی‌گیرد و خروجی لایه‌های قبلی مستقیماً به لایه‌های عمیق‌تر منتقل می‌شود. در صورت عدم استفاده از residual connection، اگر شبکه را خیلی عمیق کنیم ممکن است که شبکه حتی نتواند تابع identity را یاد بگیرد. (به دلیل ضربهای متوالی)

علاوه بر موارد ذکر شده در بالا، اضافه کردن خروجی لایه‌ی کم عمق به پرعمق تر باعث می‌شود که مقادیر بزرگ‌تری حاصل شود، مقدار مشتق ها بزرگ‌تر شود، و در اثر ضربهای متوالی گرادیان ها gradient با احتمال کمتری رخ دهد.

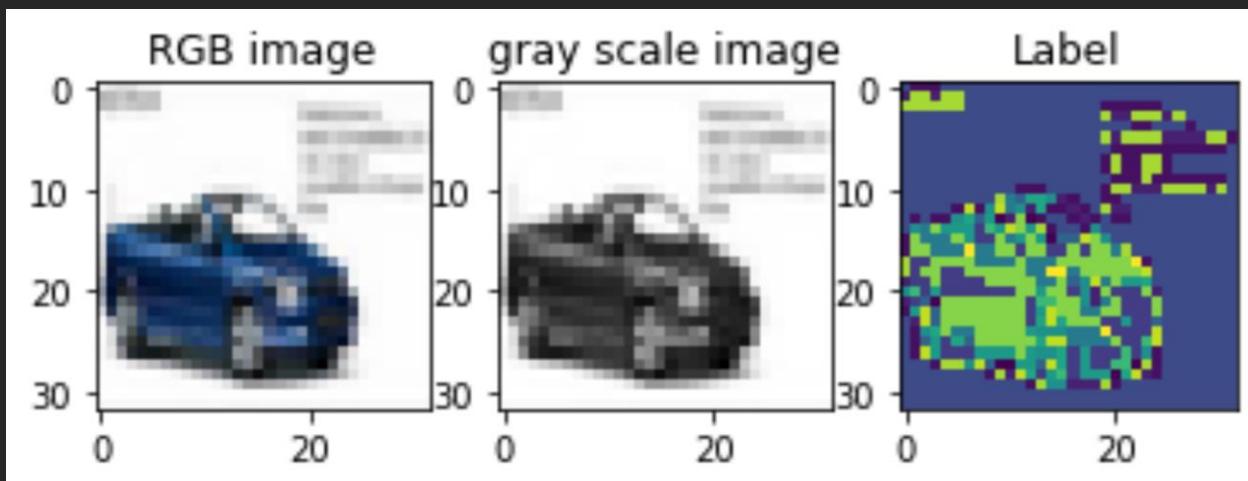
گزارش پیاده‌سازی انجام شده

در این تمرین، شبکه‌هایی با متدهای مختلف با هدف انجام مسئله image colorization پیاده‌سازی شده‌اند. این شبکه‌ها یک تصویر grayscale را به عنوان ورودی می‌گیرند. به هر کدام از پیکسل‌ها یک کلاس (در اینجا رنگ) نسبت می‌دهند و در خروجی یک تصویر رنگ شده تولید می‌کنند.

توضیح مجموعه داده

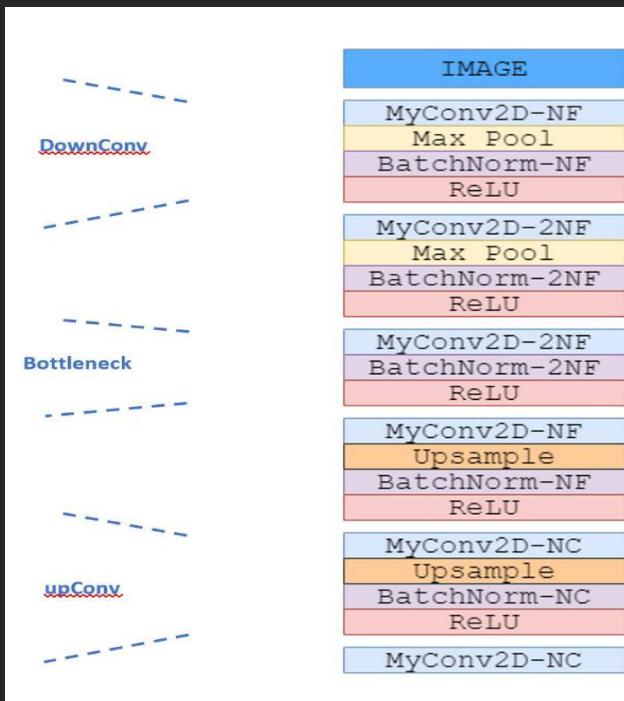
در این پروژه، از کلاس automobile از مجموعه داده cifar10 استفاده می‌شود. با انجام پیش‌پردازش‌هایی، این تصاویر RGB به تصاویر grayscale تبدیل می‌شوند و برچسب متناظر هر تصویر ورودی تولید می‌شود.

در این پروژه از 5000 تصویر برای آموزش و از 1000 تصویر برای تست استفاده می‌شود. به طور دقیق‌تر ابعاد داده‌های آموزشی و ابعاد برچسب داده‌های آموزشی $5000 \times 32 \times 32 \times 1$ است. در هر کدام از پیکسل‌های برچسب، عددی در بازه [0, 23] قرار گرفته است که نشان میدهد آن پیکسل به چه رنگی تعلق دارد. شکل 1 نمونه‌ای از تصویر RGB داده آموزش، تصویر grayscale تولید شده و برچسب متناظر آن تصویر را نشان می‌دهد.



شکل 1- نمونه تصاویر ورودی شبکه عصبی

پیاده سازی شبکه عصبی ابتدایی



شکل 2 - معماری *baseline*

در این پروژه یک شبکه عصبی ابتدایی مطابق معماری نشان داده شده در شکل 3 پیاده سازی می شود. برای پیاده سازی این شبکه سه مأذول به نام downconv و upconv و bottleneck برنامه نویسی شده است.

ماژول downconv: این ماژول به ترتیت از لایه های زیر تشکیل شده است:

1. لایه conv2d: با اعمال تعدادی فیلتر، عمل convolution روی داده ورودی این لایه انجام می شود. تعداد کانال های داده ورودی با `in_channels` و تعداد کانال های داده خروجی با `out_channels` مشخص شده است. تعداد کانال های لایه خروجی برابر تعداد فیلترهایی است که روی تصویر ورودی اعمال می کنیم. در اینجا `stride=1` و `padding=same` در نظر گرفته شده است. در نتیجه این لایه `width` و `height` داده ورودی را تغییری نمی دهد.

2. لایه Maxpool: در این لایه `Kernel-size=2` و `stride=1` در نظر گرفته شده است. در نتیجه این لایه `width` و `height` فیچرم ب پورودی را $\frac{1}{2}$ می کند. این لایه پارامتری برای آموزش ندارد.

.3. لایه BatchNorm: انجام عمل normalization روی خروجی لایه قبل که باعث normalize شدن مقادیر و سریع شدن فرآیند آموزش میشود.

.4. لایه Relu: اعمال تابع فعالیت Relu روی خروجی BatchNorm. این لایه نیز پارامتری برای یادگیری ندارد و صرفا nonlinearity به شبکه اضافه میکند.

```
class DownConv(nn.Module):
    def __init__(self, in_channels, out_channels, kernel):
        super(DownConv, self).__init__()
        self.down = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=kernel, padding='same'),
            nn.MaxPool2d(kernel_size=2),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
        )

    def forward(self, x):
        return self.down(x)
```

ماژول upconv: این ماژول به ترتیت از لایه های زیر تشکیل شده است:

.1. لایه Conv2d: توضیح آن دقیقا مانند ماژول قبلی است.

.2. لایه Upsample: با توجه به اینکه $scale-factor = 2$ قرار داده شده است، این لایه برعکس لایه maxpool در ماژول downconv که ابعاد مکانی فیچرمپ را نصف می کرد، ابعاد مکانی فیچرمپ وردی را دو برابر می کند. برای انجام نمونه افزایی به صورت دیفالت از الگوریتم nearest neighbor استفاده می شود.

.3. لایه BatchNorm: توضیح آن دقیقا مانند ماژول قبلی است.

.4. لایه Relu: توضیح آن دقیقا مانند ماژول قبلی است.

```

class UpConv(nn.Module):
    def __init__(self, in_channels, out_channels, kernel):
        super(UpConv, self).__init__()
        self.up = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=kernel, padding='same'),
            nn.Upsample(scale_factor=2),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
        )

    def forward(self, x):
        return self.up(x)

```

ماژول bottleneck: این ماژول به ترتیت از لایه‌های زیر تشکیل شده است:

1. لایه Conv2d: توضیح آن دقیقا مانند ماژول قبلی است.
2. لایه BatchNorm: توضیح آن دقیقا مانند ماژول قبلی است.
3. لایه Relu: توضیح آن دقیقا مانند ماژول قبلی است.

```

class Bottleneck(nn.Module):
    def __init__(self, in_channels, out_channels, kernel):
        super(Bottleneck, self).__init__()
        self.bottleneck = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=kernel, padding='same'),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
        )

    def forward(self, x):
        return self.bottleneck(x)

```

ساخت شبکه basemodel: برای ساخت این مدل از مازولهای پیاده‌سازی شده استفاده شده است. به ترتیب از دو مازول downconv، یک مازول bottleneck و در نهایت دو مازول upconv استفاده می‌شود. در هنگام ساخت این مازولهای کافی است به درستی تعداد کانال‌های ورودی، تعداد کانال‌های خروجی و اندازه کرنل مقدار دهی شود.

```
class BaseModel(nn.Module):
    def __init__(self, kernel, num_filters, num_colors, in_channels=1):
        super(BaseModel, self).__init__()

        self.ups = nn.ModuleList()
        self.downs = nn.ModuleList()

        self.downs.append(DownConv(in_channels, num_filters, kernel))
        self.downs.append(DownConv(num_filters, num_filters * 2, kernel))
        self.bottleneck = Bottleneck(num_filters * 2, num_filters * 2, kernel)
        self.ups.append(UpConv(num_filters * 2, num_filters, kernel))
        self.ups.append(UpConv(num_filters, num_colors, kernel))
        self.final_layer = nn.Conv2d(num_colors, num_colors, kernel, padding='same')

    def forward(self, x):
        for down in self.downs:
            x = down(x)
        x = self.bottleneck(x)
        for up in self.ups:
            x = up(x)
        x = self.final_layer(x)
        return x
```

محاسبه ابعاد فیچرمپ مشخص شده:

$$W = \frac{w - K + 2p}{s} + 1 \quad H = \frac{H - k + 2p}{s} + 1$$

After conv: $W = (32-3+2)/1 + 1 = 32$

after maxpool: $W = 32/2=16$

After conv: $H = (32-3+2)/1 + 1 = 32$

after maxpool: $H = 32/2=16$

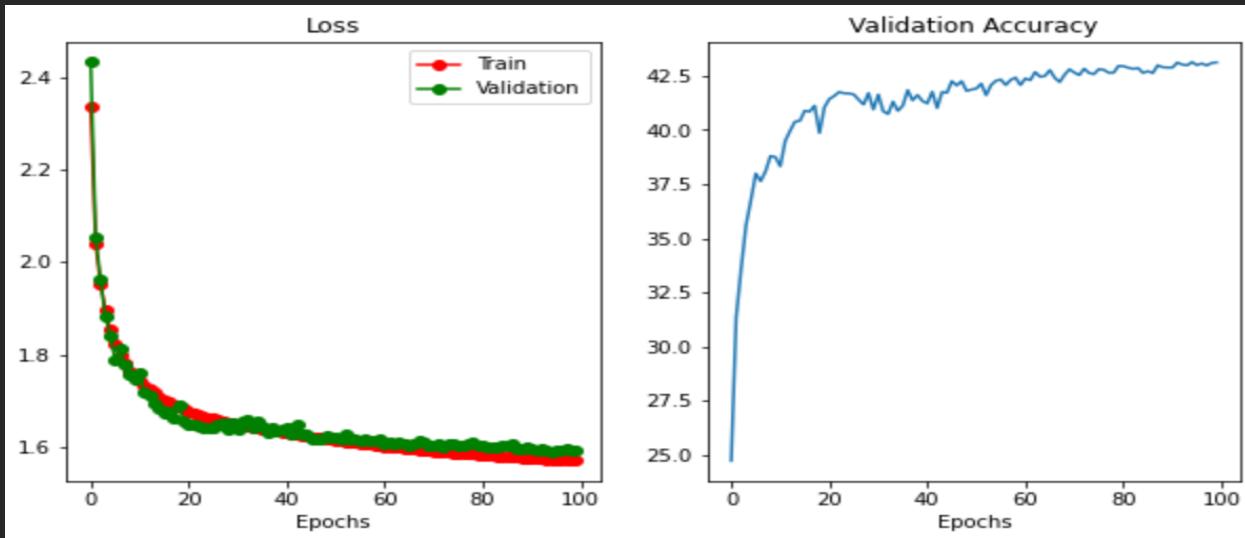
C = number of filters: 8

تنظیم پارامتر های base model

آزمایش 1- استفاده از مقدار NF های مختلف

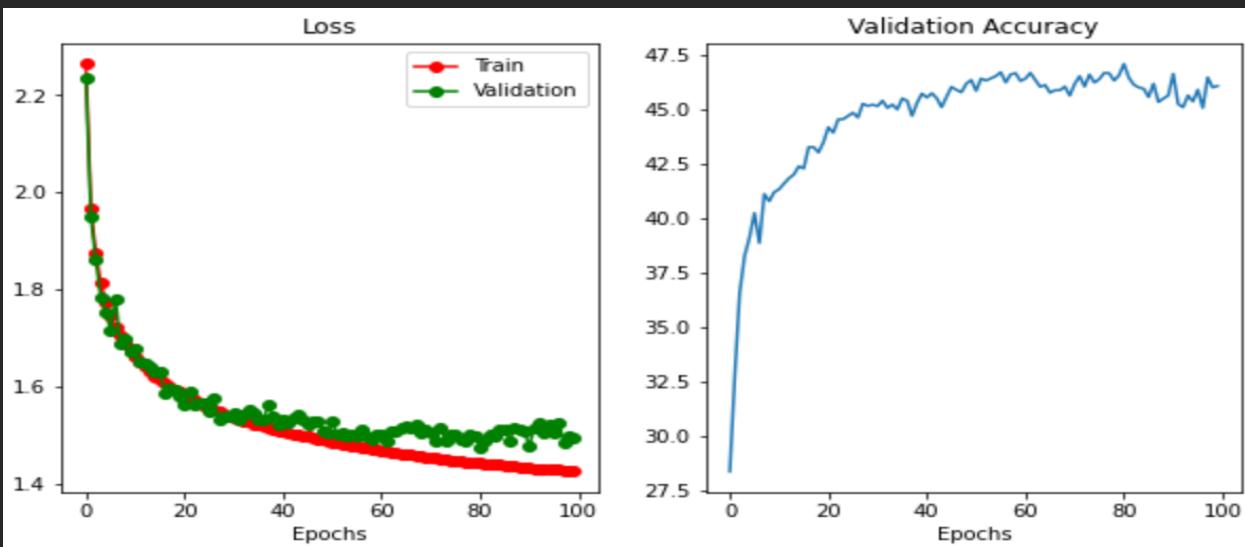
:NF=8

```
epoch: 85, train loss: 1.5789502531290054, validation loss: 1.6041726768016815, validation accuracy:42.706153869628906
epoch: 86, train loss: 1.578304386138916, validation loss: 1.6057920008897781, validation accuracy:42.62578201293945
epoch: 87, train loss: 1.577661406993866, validation loss: 1.5950284451246262, validation accuracy:42.98223114013672
epoch: 88, train loss: 1.5769392818212509, validation loss: 1.5967109501361847, validation accuracy:42.893165588378906
epoch: 89, train loss: 1.5761989504098892, validation loss: 1.6010846197605133, validation accuracy:42.87910461425781
epoch: 90, train loss: 1.5755157172679981, validation loss: 1.596976527414322, validation accuracy:42.87529754638672
epoch: 91, train loss: 1.5748321831226348, validation loss: 1.5941989719867706, validation accuracy:43.10713195800781
epoch: 92, train loss: 1.5743402808904647, validation loss: 1.5949407070875168, validation accuracy:43.01650619506836
epoch: 93, train loss: 1.5736204236745834, validation loss: 1.594417855143547, validation accuracy:42.9794921875
epoch: 94, train loss: 1.5729737430810928, validation loss: 1.590911224484437, validation accuracy:43.13154602050781
epoch: 95, train loss: 1.5724335491657258, validation loss: 1.5942815691232681, validation accuracy:42.9912109375
epoch: 96, train loss: 1.571909210085869, validation loss: 1.5933920741081238, validation accuracy:43.06904220581055
epoch: 97, train loss: 1.5714713513851166, validation loss: 1.5947080552577972, validation accuracy:42.97256088256836
epoch: 98, train loss: 1.570880365371704, validation loss: 1.5922847837209702, validation accuracy:43.08769607543945
epoch: 99, train loss: 1.5703996479511262, validation loss: 1.591906487941742, validation accuracy:43.11367416381836
```



:NF=16

```
epoch: 85, train loss: 1.438318383693695, validation loss: 1.5117734223604202, validation accuracy:45.54433822631836
epoch: 86, train loss: 1.4368693709373475, validation loss: 1.4891147762537003, validation accuracy:46.157718658447266
epoch: 87, train loss: 1.4358580261468887, validation loss: 1.5145218670368195, validation accuracy:45.31553268432617
epoch: 88, train loss: 1.4345908015966415, validation loss: 1.5121607035398483, validation accuracy:45.47676086425781
epoch: 89, train loss: 1.4335057199001313, validation loss: 1.5070190876722336, validation accuracy:45.61494445800781
epoch: 90, train loss: 1.4324674755334854, validation loss: 1.477717721261978, validation accuracy:46.610450744628906
epoch: 91, train loss: 1.4320405960083007, validation loss: 1.515474796295166, validation accuracy:45.22451400756836
epoch: 92, train loss: 1.43130304813385, validation loss: 1.5242931991815567, validation accuracy:45.07490539550781
epoch: 93, train loss: 1.431059268116951, validation loss: 1.5056924372911453, validation accuracy:45.61836242675781
epoch: 94, train loss: 1.4305506736040114, validation loss: 1.521899476647377, validation accuracy:45.34160614013672
epoch: 95, train loss: 1.4297969132661819, validation loss: 1.5052465498447418, validation accuracy:45.878028869628906
epoch: 96, train loss: 1.4290669918060304, validation loss: 1.5263393372297287, validation accuracy:45.04697799682617
epoch: 97, train loss: 1.4275770127773284, validation loss: 1.4847678542137146, validation accuracy:46.44326400756836
epoch: 98, train loss: 1.4264365494251252, validation loss: 1.4993931949138641, validation accuracy:45.98125076293945
epoch: 99, train loss: 1.4255515724420547, validation loss: 1.4959833770990372, validation accuracy:46.05361557006836
```

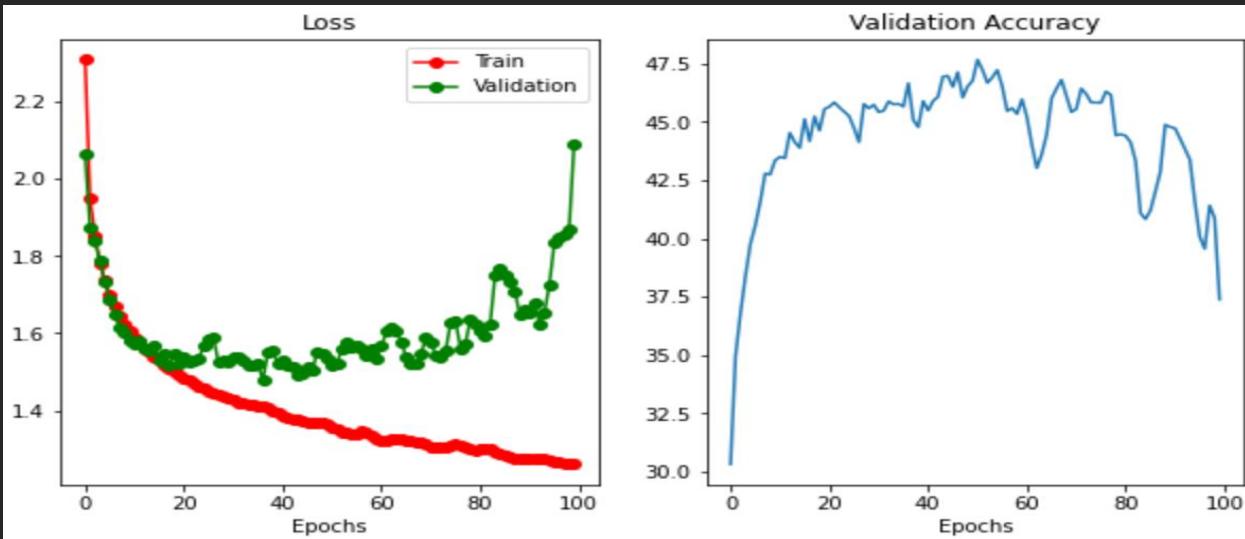


:NF=32

```

epoch: 85, train loss: 1.283780612051487, validation loss: 1.748697280883789, validation accuracy:41.198246002197266
epoch: 86, train loss: 1.2800578758120538, validation loss: 1.7317666411399841, validation accuracy:42.00820541381836
epoch: 87, train loss: 1.2773955285549163, validation loss: 1.7095473259687424, validation accuracy:42.849220275878906
epoch: 88, train loss: 1.2750179141759872, validation loss: 1.648413509130478, validation accuracy:44.87822341918945
epoch: 89, train loss: 1.2754161655902863, validation loss: 1.6596135199069977, validation accuracy:44.783203125
epoch: 90, train loss: 1.2756397247314453, validation loss: 1.6528348624706268, validation accuracy:44.72383117675781
epoch: 91, train loss: 1.2765199333429336, validation loss: 1.6796919405460358, validation accuracy:44.28603744506836
epoch: 92, train loss: 1.2765146657824515, validation loss: 1.624322310990065, validation accuracy:43.83769607543945
epoch: 93, train loss: 1.2751882553100586, validation loss: 1.6531656980514526, validation accuracy:43.38125228881836
epoch: 94, train loss: 1.2728130310773849, validation loss: 1.7259192913770676, validation accuracy:41.58378982543945
epoch: 95, train loss: 1.2690026223659516, validation loss: 1.8340086042881012, validation accuracy:40.07441711425781
epoch: 96, train loss: 1.2652679145336152, validation loss: 1.849368393421173, validation accuracy:39.55556869506836
epoch: 97, train loss: 1.261595430970192, validation loss: 1.8550743013620377, validation accuracy:41.41689682006836
epoch: 98, train loss: 1.260921910405159, validation loss: 1.8703975528478622, validation accuracy:40.88174057006836
epoch: 99, train loss: 1.2618911892175675, validation loss: 2.0915914177894592, validation accuracy:37.388771057128906

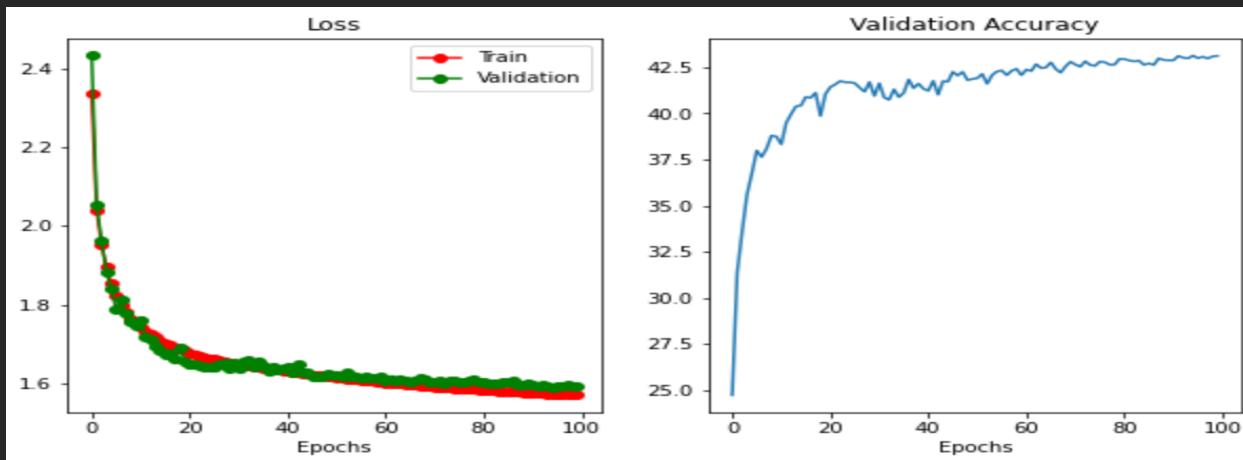
```



آزمایش 2 - استفاده از اندازه کرزل‌های مختلف

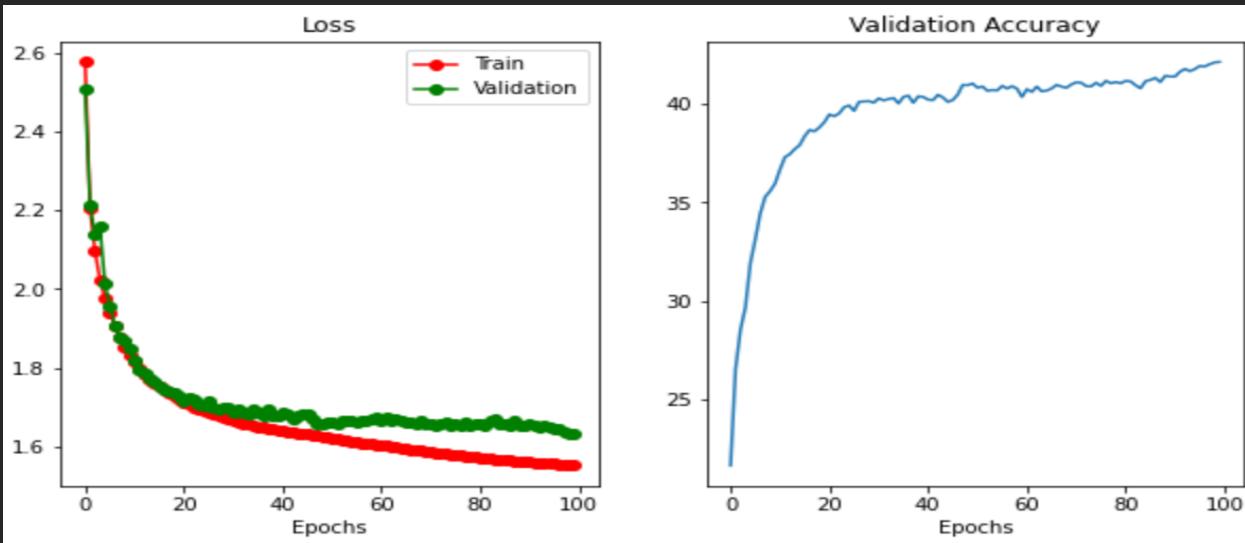
:Kernel-size=3

```
epoch: 85, train loss: 1.5789502531290054, validation loss: 1.6041726768016815, validation accuracy:42.706153869628906
epoch: 86, train loss: 1.578304386138916, validation loss: 1.6057920008897781, validation accuracy:42.62578201293945
epoch: 87, train loss: 1.577661406993866, validation loss: 1.5950284451246262, validation accuracy:42.98223114013672
epoch: 88, train loss: 1.5769392818212509, validation loss: 1.5967109501361847, validation accuracy:42.893165588378906
epoch: 89, train loss: 1.5761989504098892, validation loss: 1.6010846197605133, validation accuracy:42.87910461425781
epoch: 90, train loss: 1.5755157172679901, validation loss: 1.5969766527414322, validation accuracy:42.87529754638672
epoch: 91, train loss: 1.5748321831226348, validation loss: 1.5941989719867706, validation accuracy:43.10713195800781
epoch: 92, train loss: 1.5743402808904647, validation loss: 1.5949407070875168, validation accuracy:43.01650619506836
epoch: 93, train loss: 1.5736204236745834, validation loss: 1.594417855143547, validation accuracy:42.9794921875
epoch: 94, train loss: 1.5729737430810928, validation loss: 1.5909112244844437, validation accuracy:43.13154602050781
epoch: 95, train loss: 1.5724335491657258, validation loss: 1.5942815691232681, validation accuracy:42.9912109375
epoch: 96, train loss: 1.571909210085869, validation loss: 1.5933928741081238, validation accuracy:43.06904220581055
epoch: 97, train loss: 1.5714713513851166, validation loss: 1.5947880552577972, validation accuracy:42.97256088256836
epoch: 98, train loss: 1.570880365371704, validation loss: 1.5922847837209702, validation accuracy:43.08769607543945
epoch: 99, train loss: 1.5703996479511262, validation loss: 1.591906487941742, validation accuracy:43.11367416381836
```



:Kernel-size=5

```
epoch: 85, train loss: 1.5662127763032914, validation loss: 1.6569516062736511, validation accuracy:41.2275390625
epoch: 86, train loss: 1.5649930268526078, validation loss: 1.6537216752767563, validation accuracy:41.332035064697266
epoch: 87, train loss: 1.5639974504709244, validation loss: 1.6642022728919983, validation accuracy:41.1171875
epoch: 88, train loss: 1.5629720330238341, validation loss: 1.6554827690124512, validation accuracy:41.43545150756836
epoch: 89, train loss: 1.561846998333931, validation loss: 1.6533142626285553, validation accuracy:41.400394439697266
epoch: 90, train loss: 1.560969066619873, validation loss: 1.656690165400505, validation accuracy:41.400001525878906
epoch: 91, train loss: 1.55983846783638, validation loss: 1.6515494287014008, validation accuracy:41.64726638793945
epoch: 92, train loss: 1.559026437997818, validation loss: 1.6476188451051712, validation accuracy:41.78252410888672
epoch: 93, train loss: 1.5580947369337081, validation loss: 1.6552501916885376, validation accuracy:41.67333984375
epoch: 94, train loss: 1.5571022421121596, validation loss: 1.6510820388793945, validation accuracy:41.77900695800781
epoch: 95, train loss: 1.5560321897268294, validation loss: 1.6432950049638748, validation accuracy:41.936527252197266
epoch: 96, train loss: 1.5551148712635041, validation loss: 1.6457509249448776, validation accuracy:41.909183502197266
epoch: 97, train loss: 1.55414417386055, validation loss: 1.6380791813135147, validation accuracy:42.024906158447266
epoch: 98, train loss: 1.5530133366584777, validation loss: 1.6306934505701065, validation accuracy:42.11445617675781
epoch: 99, train loss: 1.552087625861168, validation loss: 1.63227379322052, validation accuracy:42.14179992675781
```

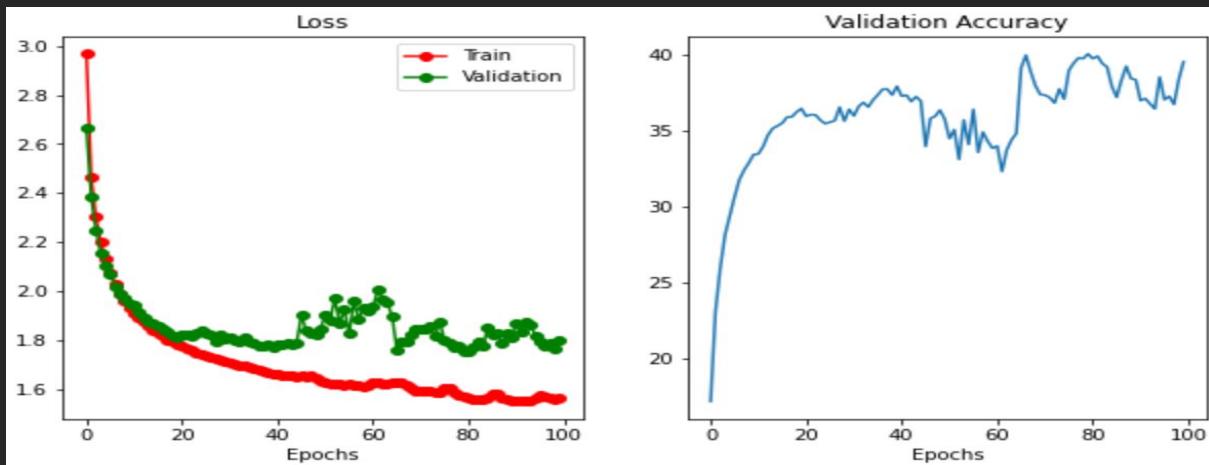


:Kernel-size=7

```

epoch: 85, train loss: 1.5824400752782821, validation loss: 1.8189221918582916, validation accuracy:37.17607498168945
epoch: 86, train loss: 1.5776596784591674, validation loss: 1.8246069848537445, validation accuracy:38.29716873168945
epoch: 87, train loss: 1.5628857463598251, validation loss: 1.7861021608114243, validation accuracy:39.218360900878906
epoch: 88, train loss: 1.5557024508714676, validation loss: 1.828594833612442, validation accuracy:38.40664291381836
epoch: 89, train loss: 1.5512076497077942, validation loss: 1.8123116940259933, validation accuracy:38.327247619628906
epoch: 90, train loss: 1.5500787526369095, validation loss: 1.865840449929374, validation accuracy:36.95869445800781
epoch: 91, train loss: 1.5499602943658828, validation loss: 1.830865278840065, validation accuracy:37.09306716918945
epoch: 92, train loss: 1.5489508390426636, validation loss: 1.8755113035440445, validation accuracy:36.7470703125
epoch: 93, train loss: 1.5539394974708558, validation loss: 1.863738313317299, validation accuracy:36.413578033447266
epoch: 94, train loss: 1.5602000772953033, validation loss: 1.8154439330101013, validation accuracy:38.503421783447266
epoch: 95, train loss: 1.5763686716556549, validation loss: 1.7935956567525864, validation accuracy:37.014747619628906
epoch: 96, train loss: 1.5668997526168824, validation loss: 1.776050791144371, validation accuracy:37.24492645263672
epoch: 97, train loss: 1.563749834895134, validation loss: 1.7887912690639496, validation accuracy:36.7080078125
epoch: 98, train loss: 1.5593135416507722, validation loss: 1.7661074846982956, validation accuracy:38.338382720947266
epoch: 99, train loss: 1.5623849779367447, validation loss: 1.7977241724729538, validation accuracy:39.50947570800781

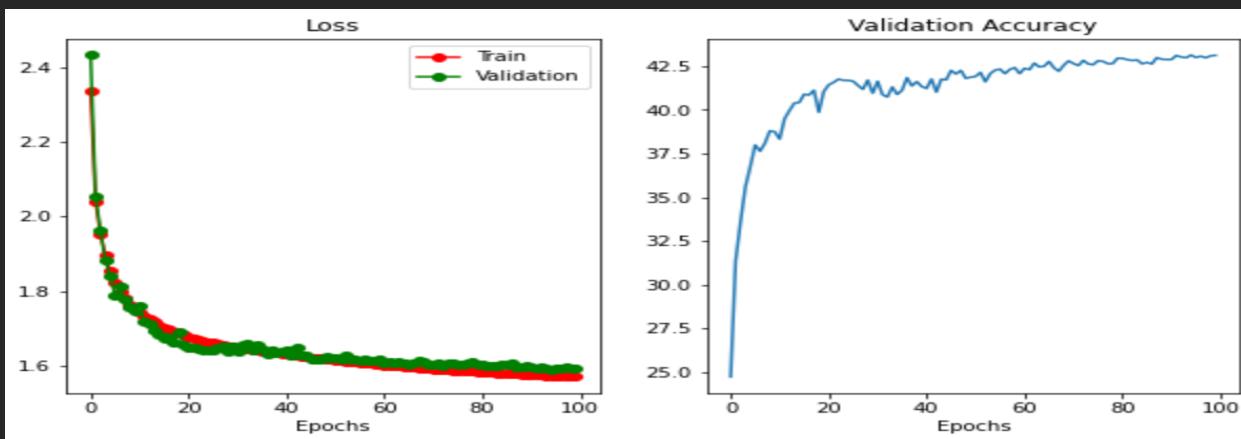
```



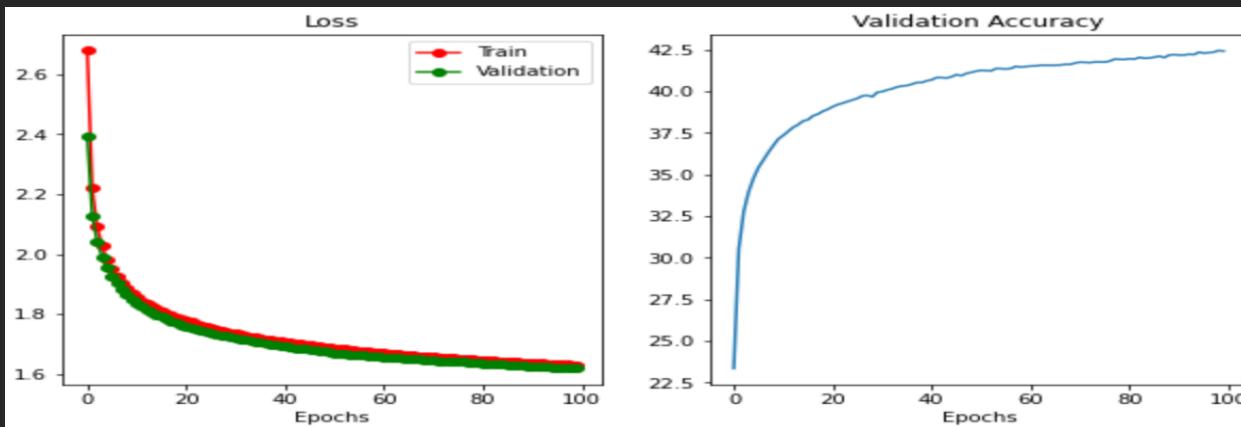
آزمایش 3 - استفاده از learning rate های مختلف

:Learning-rate=0.01

```
epoch: 85, train loss: 1.5789502531290054, validation loss: 1.6041726768016815, validation accuracy:42.706153869628906
epoch: 86, train loss: 1.578304386138916, validation loss: 1.6057920008897781, validation accuracy:42.62578201293945
epoch: 87, train loss: 1.577661406993866, validation loss: 1.595028451246262, validation accuracy:42.98223114013672
epoch: 88, train loss: 1.5769392818212509, validation loss: 1.5967109501361847, validation accuracy:42.893165588378906
epoch: 89, train loss: 1.5761989504098892, validation loss: 1.6010846197605133, validation accuracy:42.87910461425781
epoch: 90, train loss: 1.5755157172679901, validation loss: 1.5969766527414322, validation accuracy:42.87529754638672
epoch: 91, train loss: 1.5748321831226348, validation loss: 1.5941989719867706, validation accuracy:43.10713195800781
epoch: 92, train loss: 1.5743402808904647, validation loss: 1.5949407070875168, validation accuracy:43.01650619506836
epoch: 93, train loss: 1.5736204236745834, validation loss: 1.594417855143547, validation accuracy:42.9794921875
epoch: 94, train loss: 1.5729737430810928, validation loss: 1.5909112244844437, validation accuracy:43.13154602050781
epoch: 95, train loss: 1.5724335491657258, validation loss: 1.5942815691232681, validation accuracy:42.9912109375
epoch: 96, train loss: 1.571909210085869, validation loss: 1.5933920741081238, validation accuracy:43.06904220581055
epoch: 97, train loss: 1.5714713513851166, validation loss: 1.5947880552577972, validation accuracy:42.97256088256836
epoch: 98, train loss: 1.570880365371704, validation loss: 1.5922847837209702, validation accuracy:43.08769607543945
epoch: 99, train loss: 1.5703996479511262, validation loss: 1.591906487941742, validation accuracy:43.11367416381836
```



:Learning-rate=0.001



```

epoch: 85, train loss: 1.6417963624000549, validation loss: 1.630931779742241, validation accuracy:42.043949127197266
epoch: 86, train loss: 1.6408832520246506, validation loss: 1.629151239991188, validation accuracy:42.10683822631836
epoch: 87, train loss: 1.6399979084730147, validation loss: 1.6296314895153046, validation accuracy:42.01806640625
epoch: 88, train loss: 1.6391069680452346, validation loss: 1.6274648159742355, validation accuracy:42.170509338378906
epoch: 89, train loss: 1.6382503539323807, validation loss: 1.6264337003231049, validation accuracy:42.196781158447266
epoch: 90, train loss: 1.6373749822378159, validation loss: 1.6264349073171616, validation accuracy:42.168556213378906
epoch: 91, train loss: 1.6365573197603225, validation loss: 1.6258327513933182, validation accuracy:42.17119216918945
epoch: 92, train loss: 1.6356939107179642, validation loss: 1.6246047168970108, validation accuracy:42.217872619628906
epoch: 93, train loss: 1.634853821992874, validation loss: 1.624695599079132, validation accuracy:42.19472885131836
epoch: 94, train loss: 1.6340185880661011, validation loss: 1.6216742545366287, validation accuracy:42.346778869628906
epoch: 95, train loss: 1.633236327767372, validation loss: 1.622328519821167, validation accuracy:42.28144454956055
epoch: 96, train loss: 1.6324396103620529, validation loss: 1.6209022551774979, validation accuracy:42.31465148925781
epoch: 97, train loss: 1.6316727101802826, validation loss: 1.6197075843811035, validation accuracy:42.35088348388672
epoch: 98, train loss: 1.630906316637993, validation loss: 1.6187731474637985, validation accuracy:42.44179916381836
epoch: 99, train loss: 1.630131658911705, validation loss: 1.618390291929245, validation accuracy:42.411231994628906

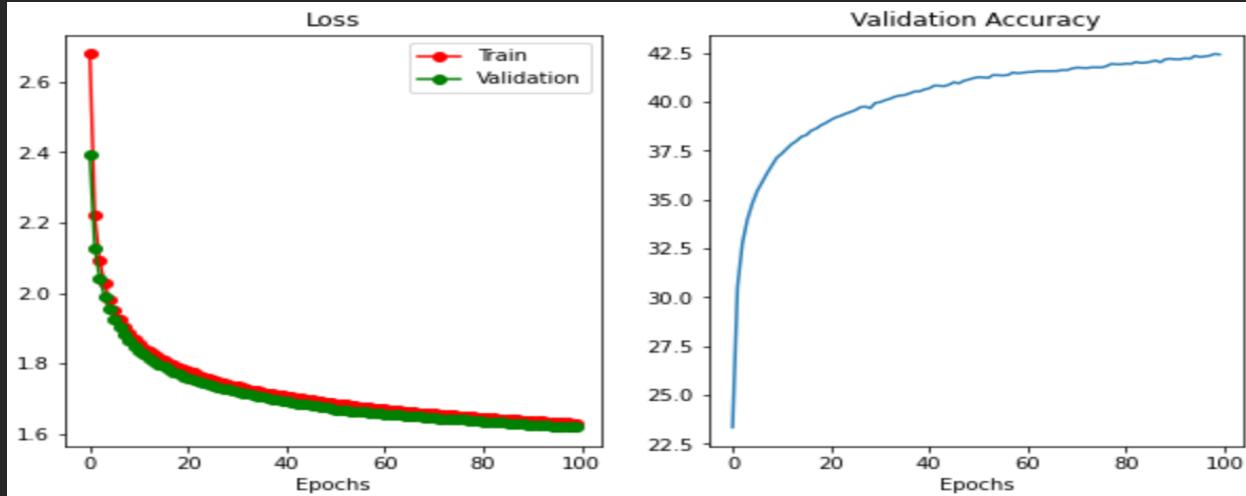
```

:Learning-rate=0.0001

```

epoch: 85, train loss: 1.8586731463670731, validation loss: 1.8427303284406662, validation accuracy:37.452735900878906
epoch: 86, train loss: 1.8573862820863725, validation loss: 1.841451108455658, validation accuracy:37.481937408447266
epoch: 87, train loss: 1.8561214357614517, validation loss: 1.840229108929634, validation accuracy:37.51445388793945
epoch: 88, train loss: 1.8548777997493744, validation loss: 1.839028388261795, validation accuracy:37.55156326293945
epoch: 89, train loss: 1.8536525279283524, validation loss: 1.8377822786569595, validation accuracy:37.5810546875
epoch: 90, train loss: 1.8524408608675003, validation loss: 1.83668052359342575, validation accuracy:37.61220932006836
epoch: 91, train loss: 1.851257038116455, validation loss: 1.8354116678237915, validation accuracy:37.63935852050781
epoch: 92, train loss: 1.8500861078500748, validation loss: 1.834266573190689, validation accuracy:37.6591796875
epoch: 93, train loss: 1.8489327073097228, validation loss: 1.8331221044063568, validation accuracy:37.69697570800781
epoch: 94, train loss: 1.8477935701608659, validation loss: 1.8319965600967407, validation accuracy:37.72822189331055
epoch: 95, train loss: 1.8466753304004668, validation loss: 1.830821231007576, validation accuracy:37.76484298706055
epoch: 96, train loss: 1.8455689787864684, validation loss: 1.8297260105609894, validation accuracy:37.786624908447266
epoch: 97, train loss: 1.8444733232259751, validation loss: 1.828693151473999, validation accuracy:37.81142807006836
epoch: 98, train loss: 1.843395119905472, validation loss: 1.8275936394929886, validation accuracy:37.8251953125
epoch: 99, train loss: 1.8423302471637726, validation loss: 1.8264575749635696, validation accuracy:37.85781478881836

```



نتیجه گیری:

پارامتر NF متناظر با تعداد فیلترهایی است که در لایه convolution استفاده خواهیم کرد. با زیاد کردن مقدار NF از تعداد فیلترهای بیشتری استفاده خواهد شد و در نتیجه تعداد پارامترهای شبکه افزایش میابد. افزایش تعداد پارامترهای به معنای پیچیده تر شدن مدل است. همانطور که میدانیم، مدل خیلی ساده دادهها را به خوبی یاد نمیگیرد و underfit میکند. اما در صورتی که مدل خیلی پیچیده باشد، روی دادههای آموزشی overfit میکند و دقت خود را نمیتواند روی داده های تست تعمیم دهد. در این تمرین مشاهده شده است که در صورتی که مقدار NF را برابر 16 یا 32 در نظر بگیریم، با وجود اینکه خطای دادههای آموزشی نزولی است، اما خطای آموزشی در هنگام آموزش قرار میگیرد و نوسان میکند. این به این معنا است که شبکه در هنگام آموزش overfit میکند و $NF=8$ به عنوان بهترین مقدار انتخاب میشود.

استفاده از kernel size های بزرگ منجر به از دست دادن جزئیات هنگام انجام پردازشها میشود. همچنین اندازه کرنل بزرگتر به معنای تعداد پارامترهای بیشتر و مدل پیچیده تر است. در اینجا مطابق با تحلیل های حالت قبلی تابع خطای مشاهده میشود در صورتی که اندازه کرنل را برابر 5 یا 7 در نظر بگیریم، Overfitting رخ می دهد.

همانطور که مطابق با تئوی میدانیم، استفاده از نرخ یادگیری خیلی کم باعث میشود که خیلی دیر به نقطه مینیمم همگرا بشویم. همچنین استفاده از نرخ یادگیری خیلی زیاد باعث واگرایی میشود. در این شبکه نرخ یادگیری 0.01 مقدار مناسبی برای آموزش شبکه است.

همچنین در صورت استفاده از بهینه ساز sgd آموزش بسیار کند انجام میشود. در این تمرین از بهینه ساز adam استفاده شده است. در صورتی که تعداد epoch ها را بزرگتر در نظر بگیریم تاثیری در عملکرد شبکه نخواهد داشت چون شبکه همگرا شده است.

اضافه کردن skip-connections

در این قسمت به شبکه پایه پیاده‌سازی شده skip connections هایی مشابه آنچه در شبکه عصبی Unet وجود دارد استفاده می‌کنیم. به گونه‌ای که خروجی لایه n و خروجی لایه یک در راستای عمق به هم متصل می‌شوند، خروجی لایه دوم و خروجی لایه $n-1$ در راستای عمق به هم متصل می‌شوند و ...

```
class CustomUNET(nn.Module):
    def __init__(self, kernel, num_filters, num_colors, in_channels=1):
        super(CustomUNET, self).__init__()

        self.first_layer = DownConv(in_channels, num_filters, kernel)
        self.second_layer = DownConv(num_filters, num_filters * 2, kernel)
        self.third_layer = Bottleneck(num_filters * 2, num_filters * 2, kernel)
        self.fourth_layer = UpConv(num_filters * 2 * 2, num_filters, kernel)
        self.fifth_layer = UpConv(num_filters * 2, num_colors, kernel)
        self.sixth_layer = nn.Conv2d(in_channels+num_colors, num_colors, kernel, padding='same')

    def forward(self, x):

        first = self.first_layer(x)
        second = self.second_layer(first)
        third = self.third_layer(second)
        fourth = self.fourth_layer(torch.cat([second, third], dim=1))
        fifth = self.fifth_layer(torch.cat([first, fourth], dim=1))
        sixth = self.sixth_layer(torch.cat([x, fifth], dim=1))

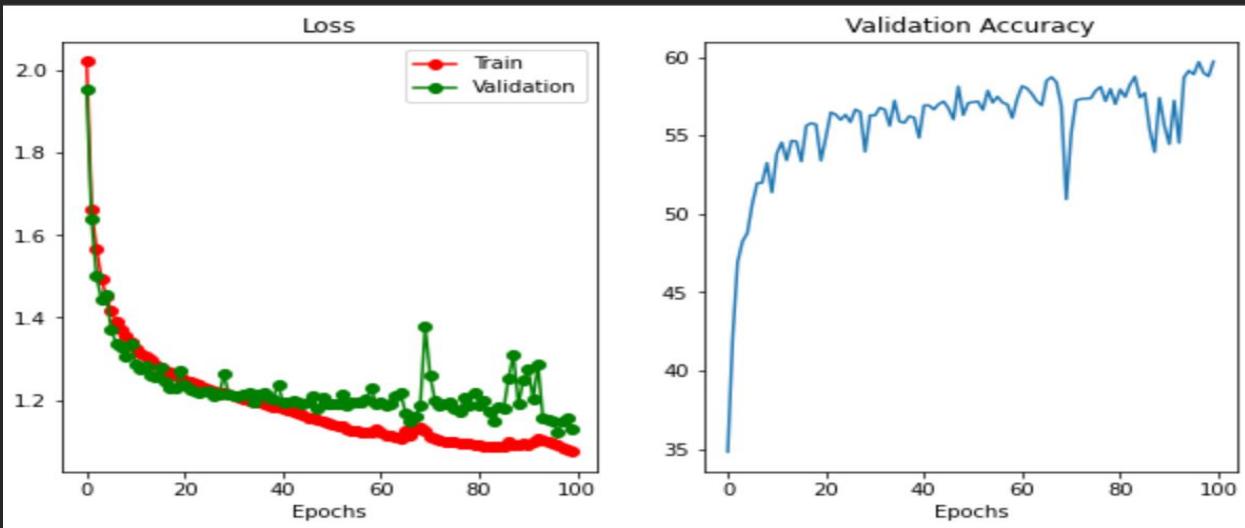
        return sixth
```

تنظیم پارامترهای :customUnet

آزمایش 1 - استفاده از مقدار NF های مختلف

:NF=8

```
epoch: 88, train loss: 1.0902841717004776, validation loss: 1.1886706203222275, validation accuracy:57.40410614013672
epoch: 89, train loss: 1.0930483728647231, validation loss: 1.2480598986148834, validation accuracy:55.619728088378906
epoch: 90, train loss: 1.091942699253559, validation loss: 1.2758862227201462, validation accuracy:54.448341369628906
epoch: 91, train loss: 1.0990852847695352, validation loss: 1.2022616118192673, validation accuracy:57.22695541381836
epoch: 92, train loss: 1.1047641411423683, validation loss: 1.2854109704494476, validation accuracy:54.526466369628906
epoch: 93, train loss: 1.1020208612084388, validation loss: 1.1559636443853378, validation accuracy:58.70000076293945
epoch: 94, train loss: 1.0968119889497756, validation loss: 1.15062116086483, validation accuracy:59.12910842895508
epoch: 95, train loss: 1.0946541339159013, validation loss: 1.1501109898090363, validation accuracy:58.89316940307617
epoch: 96, train loss: 1.0898010805249214, validation loss: 1.1231537759304047, validation accuracy:59.68223190307617
epoch: 97, train loss: 1.0813731327652931, validation loss: 1.1498697698116302, validation accuracy:58.98594284057617
epoch: 98, train loss: 1.078000046312809, validation loss: 1.1559236943721771, validation accuracy:58.76826477050781
epoch: 99, train loss: 1.073544681072235, validation loss: 1.1307340413331985, validation accuracy:59.71709442138672
```

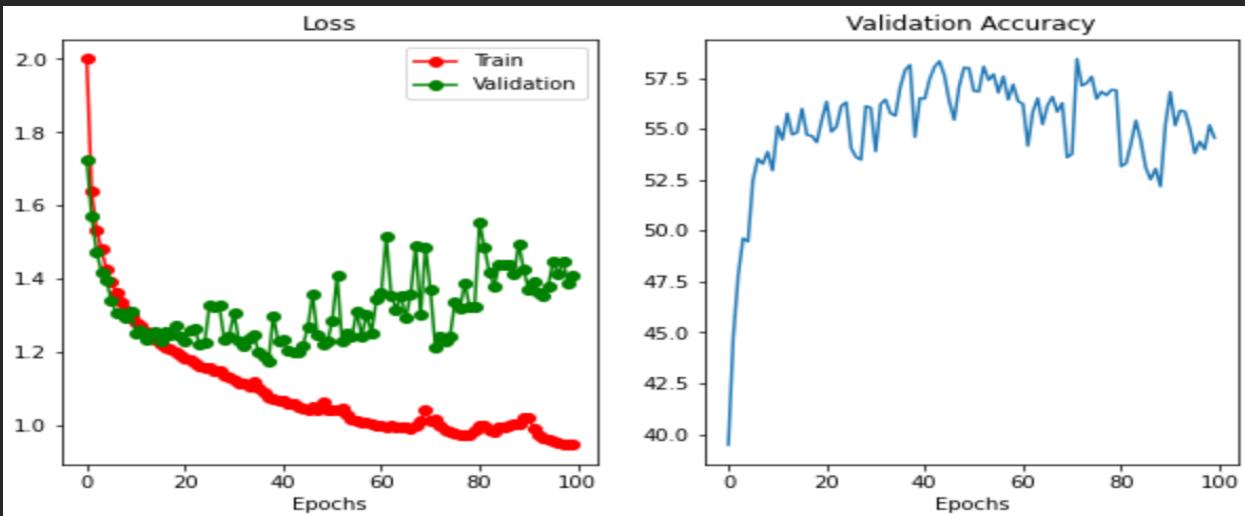


:NF=16

```

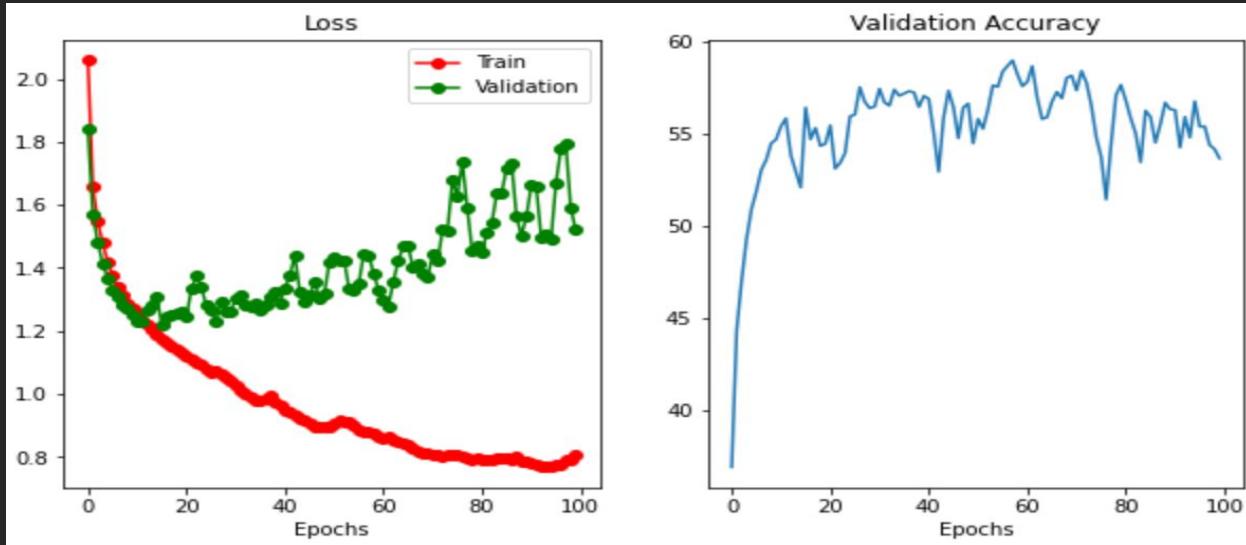
epoch: 85, train loss: 0.9934729367494584, validation loss: 1.44050133228302, validation accuracy:53.09844207763672
epoch: 86, train loss: 1.0000460922718049, validation loss: 1.4383447170257568, validation accuracy:52.50967025756836
epoch: 87, train loss: 1.0024647951126098, validation loss: 1.4122111350297928, validation accuracy:53.038578033447266
epoch: 88, train loss: 1.0030961200594901, validation loss: 1.4950926154851913, validation accuracy:52.17110061645508
epoch: 89, train loss: 1.0205057695508004, validation loss: 1.4238454103469849, validation accuracy:55.181156158447266
epoch: 90, train loss: 1.0221233695745469, validation loss: 1.3690173625946045, validation accuracy:56.81191635131836
epoch: 91, train loss: 0.9909409061074257, validation loss: 1.389933779835701, validation accuracy:55.16357421875
epoch: 92, train loss: 0.9752084235982895, validation loss: 1.3625549226999283, validation accuracy:55.900394439697266
epoch: 93, train loss: 0.9635093003511429, validation loss: 1.354966476559639, validation accuracy:55.84111785888672
epoch: 94, train loss: 0.961156365275383, validation loss: 1.380809411406517, validation accuracy:54.982421875
epoch: 95, train loss: 0.9564270183444024, validation loss: 1.4484451860189438, validation accuracy:53.793949127197266
epoch: 96, train loss: 0.9529222339391709, validation loss: 1.4116849154233932, validation accuracy:54.37646484375
epoch: 97, train loss: 0.9496283039450646, validation loss: 1.4484635144472122, validation accuracy:53.997169494628906
epoch: 98, train loss: 0.94797803068733215, validation loss: 1.3887614905834198, validation accuracy:55.180179595947266
epoch: 99, train loss: 0.9457729116082192, validation loss: 1.4088617712259293, validation accuracy:54.55733108520508

```



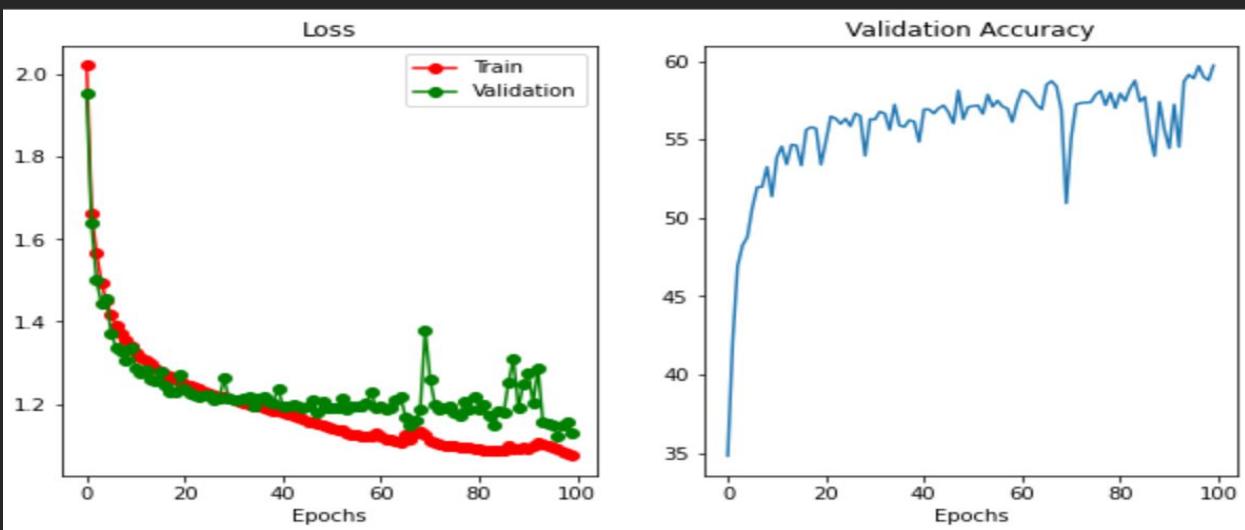
:NF=32

```
epoch: 85, train loss: 0.7947493150830269, validation loss: 1.7166669219732285, validation accuracy:55.907718658447266
epoch: 86, train loss: 0.7906655669212341, validation loss: 1.732046589255333, validation accuracy:54.500492095947266
epoch: 87, train loss: 0.8000587090849877, validation loss: 1.561829850077629, validation accuracy:55.49141311645508
epoch: 88, train loss: 0.7840755090117455, validation loss: 1.5032451301813126, validation accuracy:56.6708984375
epoch: 89, train loss: 0.7842849940061569, validation loss: 1.5629333704710007, validation accuracy:56.31221008300781
epoch: 90, train loss: 0.7791066363453865, validation loss: 1.662255808711052, validation accuracy:56.272464752197266
epoch: 91, train loss: 0.7753859132528305, validation loss: 1.6584261059761047, validation accuracy:54.23662567138672
epoch: 92, train loss: 0.7685151934623718, validation loss: 1.4990174323320389, validation accuracy:55.899322509765625
epoch: 93, train loss: 0.7722420513629913, validation loss: 1.5081181079149246, validation accuracy:54.776756286621094
epoch: 94, train loss: 0.7713086545467377, validation loss: 1.4896480292081833, validation accuracy:56.74638748168945
epoch: 95, train loss: 0.7743139103055, validation loss: 1.6684547513723373, validation accuracy:55.36474609375
epoch: 96, train loss: 0.7747583955526351, validation loss: 1.7775353491306305, validation accuracy:55.387306213378906
epoch: 97, train loss: 0.7891221940517426, validation loss: 1.7948414236307144, validation accuracy:54.374122619628906
epoch: 98, train loss: 0.7905885800719261, validation loss: 1.5912365019321442, validation accuracy:54.14922332763672
epoch: 99, train loss: 0.808728382849693, validation loss: 1.5209584087133408, validation accuracy:53.64463424682617
```



آزمایش 2 - استفاده از اندازه کرنل‌های مختلف :Kernel-size=3

```
epoch: 88, train loss: 1.0902841717004776, validation loss: 1.1886706203222275, validation accuracy:57.40410614013672
epoch: 89, train loss: 1.0930483728647231, validation loss: 1.2480598986148834, validation accuracy:55.619728088378906
epoch: 90, train loss: 1.091942699253559, validation loss: 1.2758862227201462, validation accuracy:54.448341369628906
epoch: 91, train loss: 1.0990852847695352, validation loss: 1.2022616118192673, validation accuracy:57.22695541381836
epoch: 92, train loss: 1.1047641411423683, validation loss: 1.2854109704494476, validation accuracy:54.526466369628906
epoch: 93, train loss: 1.1020208612084388, validation loss: 1.1559636443853378, validation accuracy:58.70000076293945
epoch: 94, train loss: 1.0968119889497756, validation loss: 1.15062116086483, validation accuracy:59.12910842895508
epoch: 95, train loss: 1.0946541339159013, validation loss: 1.1501109898090363, validation accuracy:58.89316940307617
epoch: 96, train loss: 1.0898010805249214, validation loss: 1.1231537759304047, validation accuracy:59.68223190307617
epoch: 97, train loss: 1.0813731327652931, validation loss: 1.1498697698116302, validation accuracy:58.98594284057617
epoch: 98, train loss: 1.078000046312809, validation loss: 1.1559236943721771, validation accuracy:58.76826477050781
epoch: 99, train loss: 1.073544681072235, validation loss: 1.1307340413331985, validation accuracy:59.71709442138672
```

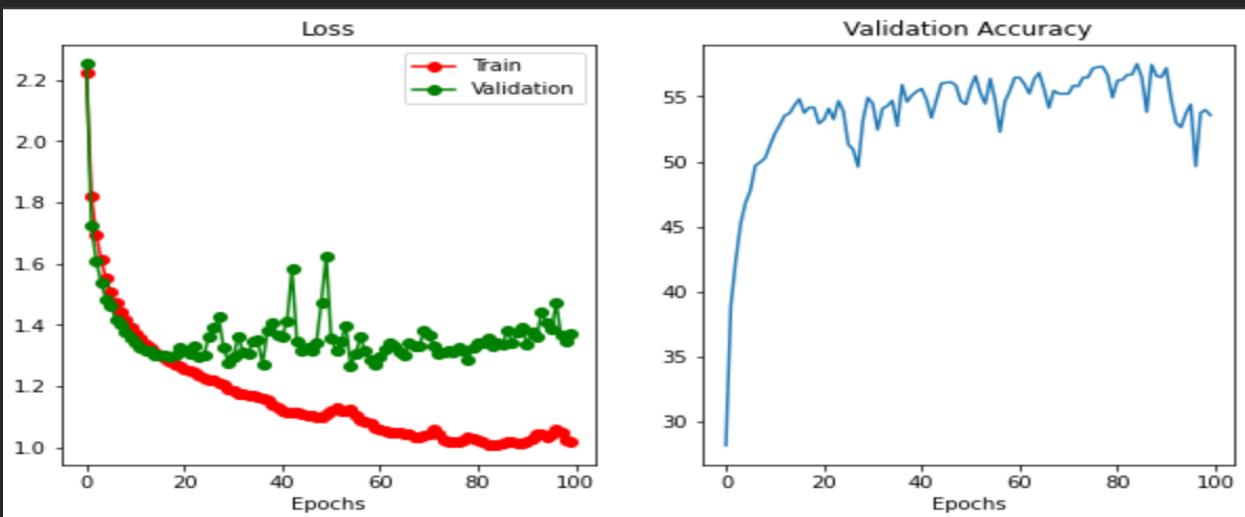


:Kernel-size=5

```

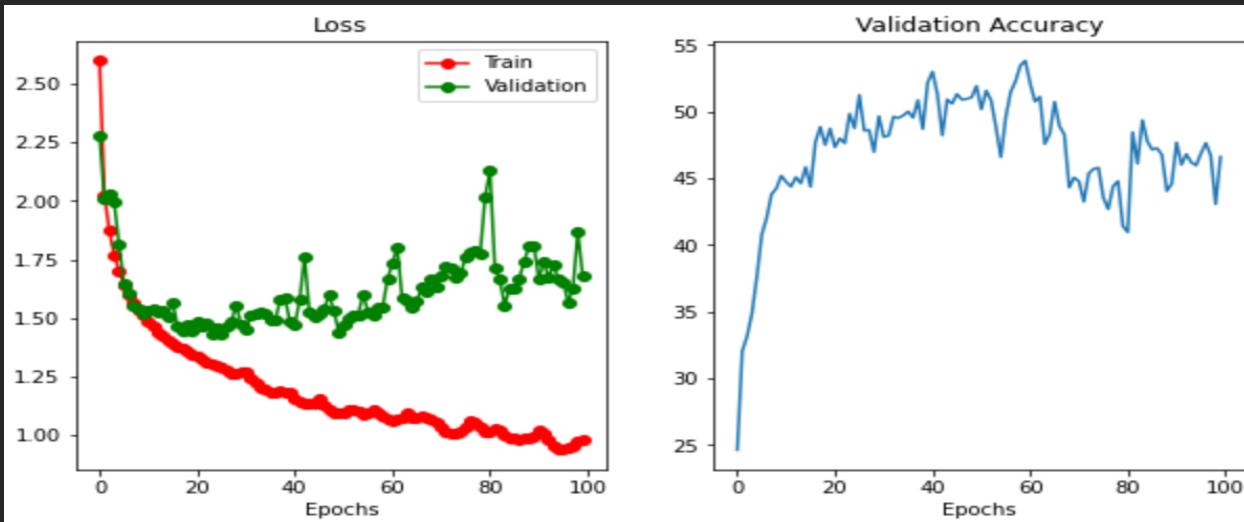
epoch: 85,  train loss: 1.0132675141096115, validation loss: 1.3364679366350174, validation accuracy:56.43926239013672
epoch: 86,  train loss: 1.0201438516378403, validation loss: 1.3801108300685883, validation accuracy:53.789649963378906
epoch: 87,  train loss: 1.0192462801933289, validation loss: 1.343013510107994, validation accuracy:57.41670227050781
epoch: 88,  train loss: 1.015750066936016, validation loss: 1.3782353699207306, validation accuracy:56.56093978881836
epoch: 89,  train loss: 1.0152266055345536, validation loss: 1.3916285783052444, validation accuracy:56.46865463256836
epoch: 90,  train loss: 1.0214343190193176, validation loss: 1.3386729955673218, validation accuracy:57.174713134765625
epoch: 91,  train loss: 1.031598335504532, validation loss: 1.3751237839460373, validation accuracy:54.76494598388672
epoch: 92,  train loss: 1.0442549616098404, validation loss: 1.3623540252447128, validation accuracy:52.97148895263672
epoch: 93,  train loss: 1.0450385436415672, validation loss: 1.4417810291051865, validation accuracy:52.621490478515625
epoch: 94,  train loss: 1.0333361104130745, validation loss: 1.4086368381977081, validation accuracy:53.6923828125
epoch: 95,  train loss: 1.0459695741534234, validation loss: 1.385554924607277, validation accuracy:54.37187576293945
epoch: 96,  train loss: 1.0568742841482162, validation loss: 1.4709258079528809, validation accuracy:49.62646484375
epoch: 97,  train loss: 1.0474839702248573, validation loss: 1.3645947575569153, validation accuracy:53.74169921875
epoch: 98,  train loss: 1.024893169105053, validation loss: 1.3469908684492111, validation accuracy:53.951568603515625
epoch: 99,  train loss: 1.0170516237616538, validation loss: 1.3740871101617813, validation accuracy:53.55068588256836

```



:Kernel-size=7

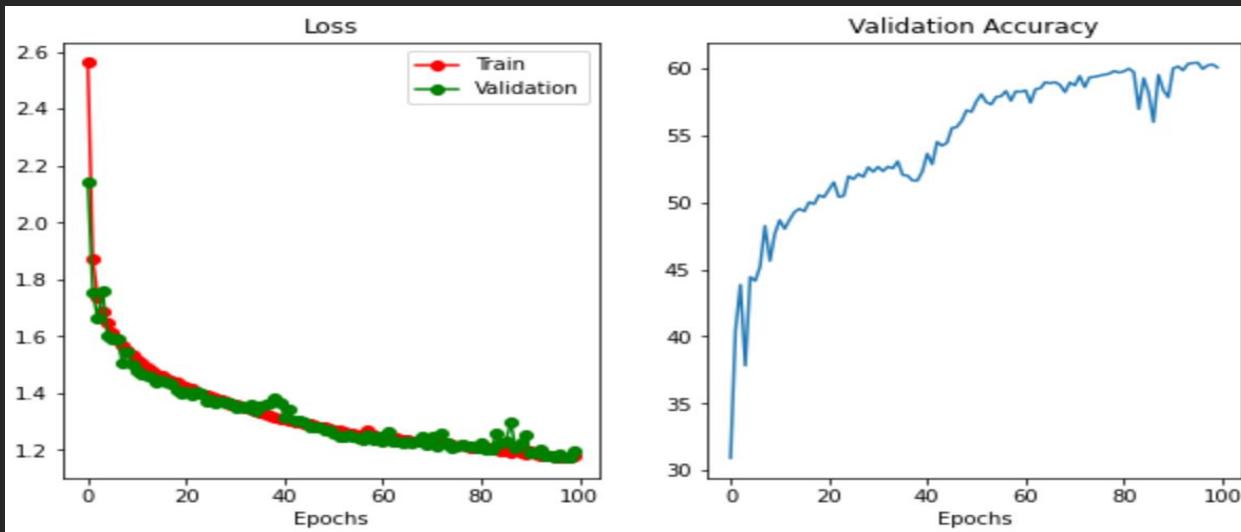
```
epoch: 85, train loss: 0.9836913734674454, validation loss: 1.62639482319355, validation accuracy:47.157814025878906
epoch: 86, train loss: 0.9819495752453804, validation loss: 1.6688572615385056, validation accuracy:47.26582336425781
epoch: 87, train loss: 0.9827962920069695, validation loss: 1.7416409850120544, validation accuracy:46.75801086425781
epoch: 88, train loss: 0.9831502318382264, validation loss: 1.8052449375391006, validation accuracy:44.051368713378906
epoch: 89, train loss: 0.9959387734532357, validation loss: 1.8102700263261795, validation accuracy:44.59511947631836
epoch: 90, train loss: 1.021645487844944, validation loss: 1.6649771481752396, validation accuracy:47.69121551513672
epoch: 91, train loss: 1.0031839609146118, validation loss: 1.739909142255783, validation accuracy:46.02412414550781
epoch: 92, train loss: 0.9814208298921585, validation loss: 1.6741646379232407, validation accuracy:46.81631088256836
epoch: 93, train loss: 0.9506645187735557, validation loss: 1.7240804880857468, validation accuracy:46.20351791381836
epoch: 94, train loss: 0.936396786570549, validation loss: 1.6679724752902985, validation accuracy:45.96553039550781
epoch: 95, train loss: 0.9357247665524483, validation loss: 1.6525249630212784, validation accuracy:46.87832260131836
epoch: 96, train loss: 0.9444225639104843, validation loss: 1.566867619752884, validation accuracy:47.65508270263672
epoch: 97, train loss: 0.9522351145744323, validation loss: 1.6257042586803436, validation accuracy:46.72500228881836
epoch: 98, train loss: 0.9755735203623772, validation loss: 1.8687705844640732, validation accuracy:43.06787109375
epoch: 99, train loss: 0.9788795858621597, validation loss: 1.677188217639923, validation accuracy:46.62236404418945
```



آزمایش 3 - استفاده از learning rate های مختلف

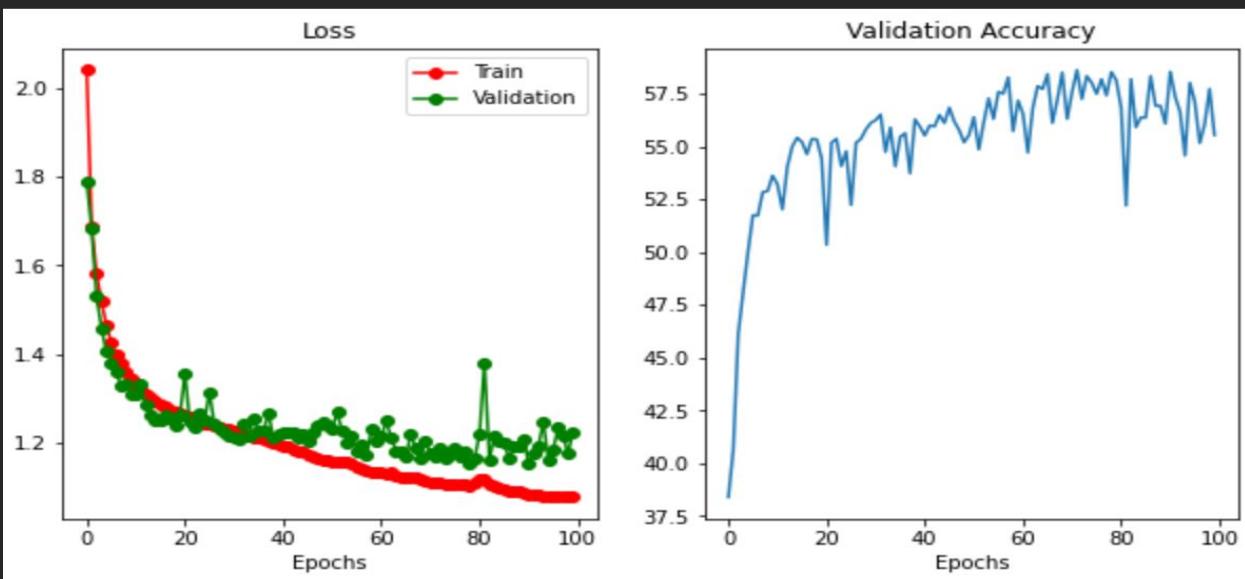
:Learning-rate=0.1

```
epoch: 85, train loss: 1.1959249258041382, validation loss: 1.2332365810871124, validation accuracy:58.18242263793945
epoch: 86, train loss: 1.1921670615673066, validation loss: 1.2958462089300156, validation accuracy:55.996681213378906
epoch: 87, train loss: 1.1992881149053574, validation loss: 1.2059055268764496, validation accuracy:59.51533508300781
epoch: 88, train loss: 1.1897992849349976, validation loss: 1.2255025655031204, validation accuracy:58.375885009765625
epoch: 89, train loss: 1.1869406253099442, validation loss: 1.2503501623868942, validation accuracy:57.84502029418945
epoch: 90, train loss: 1.1949981480836869, validation loss: 1.1931338608264923, validation accuracy:60.001468658447266
epoch: 91, train loss: 1.184665207624436, validation loss: 1.1933721154928207, validation accuracy:60.14463424682617
epoch: 92, train loss: 1.178553867340088, validation loss: 1.204550102353096, validation accuracy:59.8515625
epoch: 93, train loss: 1.1867289155721665, validation loss: 1.182454138994217, validation accuracy:60.337501525878906
epoch: 94, train loss: 1.1825680404901504, validation loss: 1.1802870631217957, validation accuracy:60.39453125
epoch: 95, train loss: 1.174422213435173, validation loss: 1.1752597093582153, validation accuracy:60.43183898925781
epoch: 96, train loss: 1.1739090204238891, validation loss: 1.186852127313614, validation accuracy:59.958595275878906
epoch: 97, train loss: 1.1716590911149978, validation loss: 1.1755444705486298, validation accuracy:60.22139358520508
epoch: 98, train loss: 1.1716723561286926, validation loss: 1.1754203736782074, validation accuracy:60.299224853515625
epoch: 99, train loss: 1.1775099903345108, validation loss: 1.195421040058136, validation accuracy:60.06631088256836
```



:Learning-rate=0.01

```
epoch: 85, train loss: 1.0930626779794692, validation loss: 1.1998246759176254, validation accuracy:56.383792877197266
epoch: 86, train loss: 1.0921310842037202, validation loss: 1.1628352403640747, validation accuracy:58.354496002197266
epoch: 87, train loss: 1.090850266814232, validation loss: 1.1902131140232086, validation accuracy:56.94521713256836
epoch: 88, train loss: 1.0884964019060135, validation loss: 1.1917406916618347, validation accuracy:56.94267654418945
epoch: 89, train loss: 1.0863746732473374, validation loss: 1.2069606184959412, validation accuracy:56.09658432006836
epoch: 90, train loss: 1.0840803414583207, validation loss: 1.1524032801389694, validation accuracy:58.561038970947266
epoch: 91, train loss: 1.082411703467369, validation loss: 1.1768961548805237, validation accuracy:57.37148666381836
epoch: 92, train loss: 1.0807879120111465, validation loss: 1.1922259032726288, validation accuracy:56.651371002197266
epoch: 93, train loss: 1.079941739141941, validation loss: 1.2471353113651276, validation accuracy:54.58193588256836
epoch: 94, train loss: 1.0788569539785384, validation loss: 1.1599715650081635, validation accuracy:58.036720275878906
epoch: 95, train loss: 1.078255297243595, validation loss: 1.184597596526146, validation accuracy:57.14101791381836
epoch: 96, train loss: 1.077618832886219, validation loss: 1.2328908741474152, validation accuracy:55.17734909057617
epoch: 97, train loss: 1.077127182483673, validation loss: 1.216034322977066, validation accuracy:56.06533432006836
epoch: 98, train loss: 1.076748864352703, validation loss: 1.175295427441597, validation accuracy:57.74619674682617
epoch: 99, train loss: 1.0773172199726104, validation loss: 1.2210049331188202, validation accuracy:55.55430221557617
```

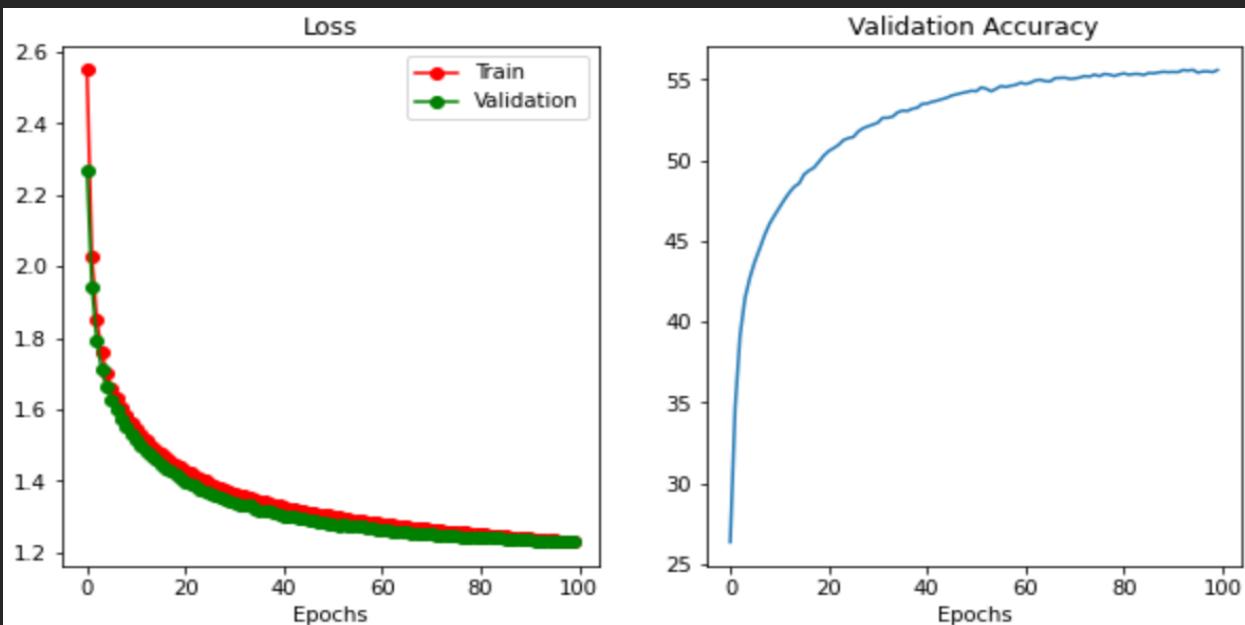


:Learning-rate=0.001

```

epoch: 85, train loss: 1.2446404486894607, validation loss: 1.2363993674516678, validation accuracy:55.40879440307617
epoch: 86, train loss: 1.243354818224907, validation loss: 1.2363749891519547, validation accuracy:55.388675689697266
epoch: 87, train loss: 1.2422424018383027, validation loss: 1.235614612698555, validation accuracy:55.44258499145508
epoch: 88, train loss: 1.2410588085651397, validation loss: 1.2344568073749542, validation accuracy:55.485355377197266
epoch: 89, train loss: 1.2399473160505294, validation loss: 1.234193667769432, validation accuracy:55.46133041381836
epoch: 90, train loss: 1.23891684114933, validation loss: 1.23356005549438085, validation accuracy:55.47568893432617
epoch: 91, train loss: 1.2379428684711455, validation loss: 1.2331628054380417, validation accuracy:55.46308898925781
epoch: 92, train loss: 1.2369162559509277, validation loss: 1.2309459298849106, validation accuracy:55.59346008300781
epoch: 93, train loss: 1.235805094242096, validation loss: 1.231423705816269, validation accuracy:55.55781555175781
epoch: 94, train loss: 1.2347776502370835, validation loss: 1.2304955124855042, validation accuracy:55.60429763793945
epoch: 95, train loss: 1.2337006032466888, validation loss: 1.2324427664279938, validation accuracy:55.42890930175781
epoch: 96, train loss: 1.2326752454042436, validation loss: 1.230938732624054, validation accuracy:55.50634765625
epoch: 97, train loss: 1.2317289799451827, validation loss: 1.2304208725690842, validation accuracy:55.51152801513672
epoch: 98, train loss: 1.2305259913206101, validation loss: 1.2304516285657883, validation accuracy:55.46348190307617
epoch: 99, train loss: 1.2295255005359649, validation loss: 1.2281605303287506, validation accuracy:55.593360900878906

```



نتیجه گیری: پارامتر NF متناظر با تعداد فیلترهایی است که در لایه convolution استفاده خواهیم کرد. با زیاد کردن مقدار NF از تعداد فیلترهای بیشتری استفاده خواهد شد و در نتیجه تعداد پارامترهای شبکه افزایش میابد. افزایش تعداد پارامترها به معنای پیچیده‌تر شدن مدل است. همانطور که میدانیم در صورتی که مدل خیلی ساده باشد، داده‌ها را به خوبی یاد نمی‌گیرد و underfit میکند. در صورتی که مدل خیلی پیچیده باشد، مدل روی داده‌های آموزشی overfit میکند و دقت خود را نمیتواند روی داده‌های تست تعمیم دهد. در این تمرین مشاهده شده که در صورت افزایش تعداد فیلترها، خطای آموزش همچنان نزولی باقی می‌ماند. یعنی مدل داده‌های آموزشی را به خوبی یاد می‌گیرد. اما وقتی که تعداد فیلترها را 16 و 32 قرار میدهیم، میبینیم که خطای validation بالاتر از خطای train قرار می‌گیرد و نوسان می‌کند. این به این معنا است که مدل Overfit میکند. بنابراین $NF=8$ به عنوان بهترین مقدار انتخاب می‌شود.

استفاده از kernel size های بزرگ منجر به از دست دادن جزئیات هنگام انجام پردازش‌ها می‌شود. همچنین استفاده از اندازه کرنل بزرگتر به معنای استفاده از تعداد پارامترهای بیشتر است. در صورت استفاده از kernel size های 5 و 7، پیچیدگی مدل زیاد می‌شود و overfitting رخ می‌دهد. چون با وجود اینکه خطای آموزش نزولی است، خطای validation بالاتر از خطای train قرار می‌گیرد و نوسان می‌کند.

در ادامه نرخ‌های آموزش مختلف بررسی شده است. در صورتی که مقدار نرخ آموزش عدد کوچکی باشد، آموزش کند تر انجام می‌شود. در صورت بزرگ بودن نرخ آموزش، ممکن است نقطه overshoot را minimum کند. در اینجا نرخ آموزش 0.1 مقدار مناسبی تشخیص داده شده است.

همچنین در صورت استفاده از بهینه ساز sgd آموزش بسیار کند انجام می‌شود. در این تمرین از بهینه ساز adam استفاده شده است. در صورتی که تعداد epoch ها را بزرگتر در نظر بگیریم تاثیری در عملکرد شبکه نخواهد داشت چون شبکه همگرا شده است.

اضافه کردن residual blocks

در این قسمت از پروژه، مازول‌های bottleneck و upconv، downconv می‌شوند. برای اینکار ورودی این مازول‌ها با خروجی مازول residual block جمع‌زده می‌شود. اما قبل از انجام ابعاد دو فیچرمپ با استفاده از element-wise لایه‌های convolution یا upsample با هم برابر می‌شوند. معماری شبکه تغییر یافته در قطعه کد زیر نشان داده شده است:

```
class CustomUnetWithResiduals(nn.Module):
    def __init__(self, kernel, num_filters, num_colors, in_channels=1):
        super(CustomUnetWithResiduals, self).__init__()

        self.first_layer = DownConv(in_channels, num_filters, kernel)
        self.skip_first_layer = SkipConnection(in_channels, num_filters, enc_or_dec='encoder')
        self.second_layer = DownConv(num_filters, num_filters * 2, kernel)
        self.skip_second_layer = SkipConnection(num_filters, num_filters * 2, enc_or_dec='encoder')
        self.third_layer = Bottleneck(num_filters * 2, num_filters * 2, kernel)
        self.fourth_layer = UpConv(num_filters * 2 * 2, num_filters, kernel)
        self.skip_fourth_layer = SkipConnection(num_filters * 2 * 2, num_filters, enc_or_dec='decoder')
        self.fifth_layer = UpConv(num_filters * 2, num_colors, kernel)
        self.skip_fifth_layer = SkipConnection(num_filters * 2, num_colors, enc_or_dec='decoder')
        self.sixth_layer = nn.Conv2d(in_channels + num_colors, num_colors, kernel, padding='same')

    def forward(self, x):
        first = self.first_layer(x)
        skip = self.skip_first_layer(x)
        first = first + skip
        second = self.second_layer(first)
        skip = self.skip_second_layer(first)
        second = second + skip
        third = self.third_layer(second)
        third = third + second
        fourth = self.fourth_layer(torch.cat([second, third], dim=1))
        skip = self.skip_fourth_layer(torch.cat([second, third], dim=1))
        fourth = fourth + skip
        fifth = self.fifth_layer(torch.cat([first, fourth], dim=1))
        skip = self.skip_fifth_layer(torch.cat([first, fourth], dim=1))
        fifth = fifth + skip
        sixth = self.sixth_layer(torch.cat([x, fifth], dim=1))

    return sixth
```

```

class SkipConnection(nn.Module):
    def __init__(self, in_channels, out_channels, enc_or_dec='encoder'):
        super(SkipConnection, self).__init__()
        self.enc_or_dec = enc_or_dec
        self.skip_encoder = nn.Sequential(
            nn.Conv2d(in_channels=in_channels, out_channels=out_channels, stride=2, kernel_size=1),
            nn.BatchNorm2d(out_channels)
        )

        self.skip_decoder = nn.Sequential(
            nn.Upsample(scale_factor=2),
            nn.Conv2d(in_channels=in_channels, out_channels=out_channels, kernel_size=1),
            nn.BatchNorm2d(out_channels)
        )

    def forward(self, x):
        if self.enc_or_dec == 'encoder':
            return self.skip_encoder(x)
        if self.enc_or_dec == 'decoder':
            return self.skip_decoder(x)

```

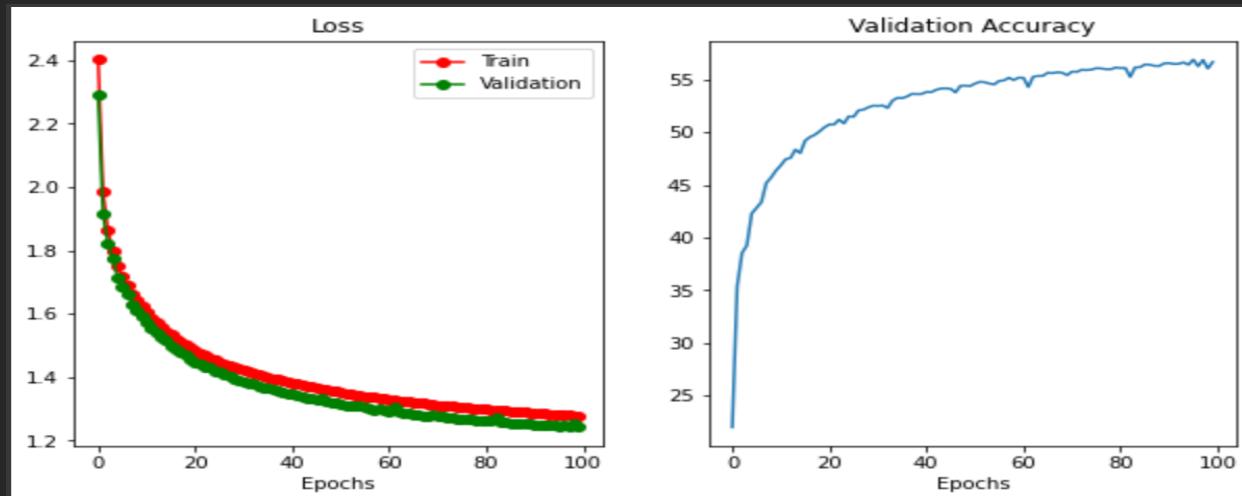
خروجی آموزش شبکه حاوی residual blocks

در نهایت این شبکه با kernel_size=3، learning rate=0.001، NF=4 بهترین نتیجه را میدهد.

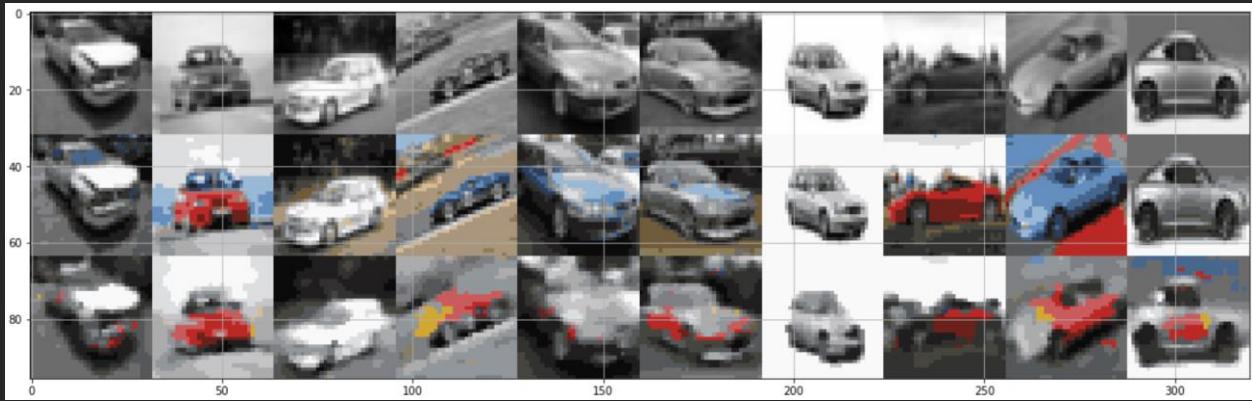
```

epoch: 85, train loss: 1.2926416248083115, validation loss: 1.254771739244461, validation accuracy:56.42197799682617
epoch: 86, train loss: 1.2915675669908524, validation loss: 1.2544443607330322, validation accuracy:56.404693603515625
epoch: 87, train loss: 1.2905131548643112, validation loss: 1.2550488114356995, validation accuracy:56.298828125
epoch: 88, train loss: 1.2894556760787963, validation loss: 1.2547224164000994, validation accuracy:56.28184127807617
epoch: 89, train loss: 1.2884186238050461, validation loss: 1.2514096647500992, validation accuracy:56.529884338378906
epoch: 90, train loss: 1.2873772233724594, validation loss: 1.2504370361566544, validation accuracy:56.55742263793945
epoch: 91, train loss: 1.2863175690174102, validation loss: 1.2508358806371689, validation accuracy:56.46650695800781
epoch: 92, train loss: 1.2851745009422302, validation loss: 1.2494235187768936, validation accuracy:56.48760223388672
epoch: 93, train loss: 1.2840773344039917, validation loss: 1.2471762001514435, validation accuracy:56.62910461425781
epoch: 94, train loss: 1.282989177107811, validation loss: 1.24966461956501, validation accuracy:56.39219284057617
epoch: 95, train loss: 1.2819061189889909, validation loss: 1.243583396877156, validation accuracy:56.88955307006836
epoch: 96, train loss: 1.280857762694359, validation loss: 1.2502038925886154, validation accuracy:56.253807067871094
epoch: 97, train loss: 1.279813939332962, validation loss: 1.2421560287475586, validation accuracy:56.85908126831055
epoch: 98, train loss: 1.2787756115198134, validation loss: 1.2524569928646088, validation accuracy:56.039554595947266
epoch: 99, train loss: 1.2777654469013213, validation loss: 1.2432546764612198, validation accuracy:56.66396713256836

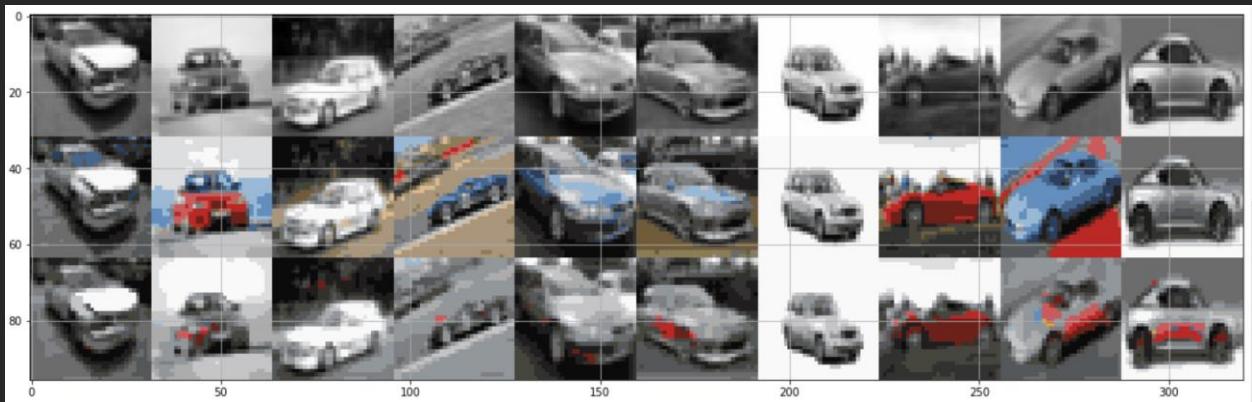
```



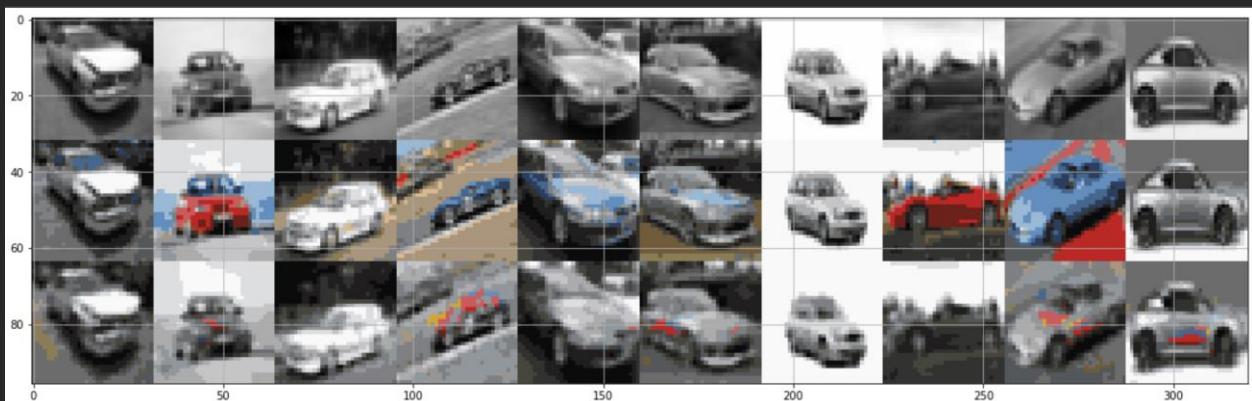
خروجی مدل ها:



خروجی *basemodel*



خروجی *customUNET*



خروجی *customUNET with residual blocks*