به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین اول

پرسش ۱	نام و نام خانوادگی	علی قربانی برگانی
	شماره دانشجویی	۸۱۰۱۰۳۲۰۹
پرسش ۲	نام و نام خانوادگی	مبین تیرافکن
	شماره دانشجویی	۸۱۰۱۰۳۰۹۱
	مهلت ارسال پاسخ	۱۴۰۱.۰۷.۰۱

فهرست اصلی:

فهرست شکل ها:

فهرست جداول

فهرست فرمول ها

فهرست نمودار ها

پرسش۱ – سگمنتیشن تصاویر شهری

۱-۱. توصیف مدل ارائه شده

3.2.1 Learning to Downsample

In our learning to downsample module, we employ three layers. Only three layers are employed to ensure low-level feature sharing is valid, and efficiently implemented. The first layer is a standard convolutional layer (Conv2D) and the remaining two layers are depthwise separable convolutional layers (DSConv). Here we emphasize, although DSConv is computationally more efficient, we employ Conv2D since the input image only has three channels, making DSConv's computational benefit insignificant at this stage.

تصویر ۱ - بخش **learning to downsample** مقاله مرجع

در ماژول Learning to Downsample دقیقا سه تا لایه داریم:

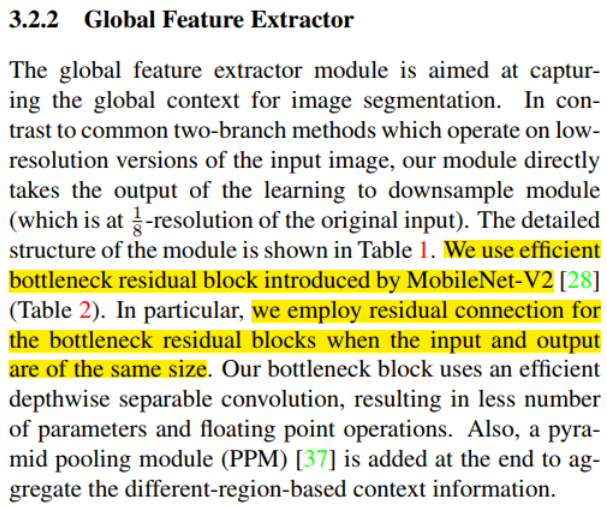
- **Conv2D**

لایه‌ی اول یک standard convolutional layer (Conv2D) است. چون ورودی فقط ۳ کانال (RGB)داره، استفاده از DSConv اینجا فایده‌ی محاسباتی زیادی نداره، پس Conv2D ساده‌تر و بهینه‌تر است.

- دو تا **DSConv**

لایه‌های دوم و سوم هر دو depthwise separable convolution که مخففش DSConv هست هستند .هدف‌شون کاهش هزینه‌های محاسباتی در استخراج ویژگی‌های ساده-low level features از تصویر است.

نکته‌ی مهم این است که فقط همین سه لایه به اشتراک‌گذاری ویژگی‌های low-level بین مسیرهای مختلف detail و context کمک می‌کند و در عین حال اجرای سریع و کم‌حجم رو تضمین می‌کند.



3.2.2 Global Feature Extractor

The global feature extractor module is aimed at capturing the global context for image segmentation. In contrast to common two-branch methods which operate on low-resolution versions of the input image, our module directly takes the output of the learning to downsample module (which is at $\frac{1}{8}$-resolution of the original input). The detailed structure of the module is shown in Table 1. We use efficient bottleneck residual block introduced by MobileNet-V2 [28] (Table 2). In particular, we employ residual connection for the bottleneck residual blocks when the input and output are of the same size. Our bottleneck block uses an efficient depthwise separable convolution, resulting in less number of parameters and floating point operations. Also, a pyramid pooling module (PPM) [37] is added at the end to aggregate the different-region-based context information.

تصویر ۲ - بخش **Global feature extractor** مقاله مرجع

در ماژول Global Feature Extractor کار به این صورت است:

۱. ورودی این ماژول، خروجی ماژول Learning to Downsample است (رزولوشن ۸/۱ تصویر اصلی).

۲. به‌جای استفاده از دو شاخه‌ی مجزا، اینجا یک سری efficient bottleneck residual block از موبایل‌نت‌وی۲ (MobileNet-V2) قرار گرفته.

- هر بلاک شامل یک مقداردهی اولیه (expansion) با کانولوشن نقطه‌ای، بعد یک depthwise separable convolution و نهایتاً یک کانولوشن نقطه‌ای دیگر است.
- residual connection فقط زمانی فعال می‌شود که اندازه و تعداد کانال‌های ورودی و خروجی بلاک یکسان باشند. این کار هم به تثبیت جریان گرادیان کمک می‌کند و هم دقت را حفظ می‌کند.

۳. در انتهای این زنجیره، یک Pyramid Pooling Module (PPM) اضافه می‌شود تا بتوان اطلاعات زمینه‌ای از مقیاس‌های مختلف را جمع‌آوری (aggregate) کرد. هر سطح از PPM تصویر را به شبکه‌های کوچک‌تر شبکه‌بندی (grid) تقسیم می‌کند، pooling انجام می‌دهد و خروجی‌ها را concate می‌کند تا ببینیم چه ویژگی‌هایی در مناطق گوناگون تصویر مهم‌اند.

این طراحی باعث می‌شود تا با کمترین پارامتر و کمترین FLOPs هم زمینه کلی (global context) تصویر استخراج شود و هم سرعت مدل حفظ شود.

3.2.3 Feature Fusion Module

Similar to ICNet [36] and ContextNet [21] we prefer simple addition of the features to ensure efficiency. Alternatively, more sophisticated feature fusion modules (*e.g.* [34]) could be employed at the cost of runtime performance, to reach better accuracy. The detail of the feature fusion module is shown in Table 3.

تصویر ۳ - بخش **feature fusion module** مقاله مرجع

در بخش3.2.3 Feature Fusion Modul مدل Fast-SCNN ، نویسنده‌ها تاکید می‌کنند که برای حفظ efficiency از یک simple addition بین دو شاخه‌ی feature استفاده می‌کنند:

۱. از خروجی شاخه‌ی detail (رزولوشن بالاتر) و خروجی شاخه‌ی context (رزولوشن پایین‌تر) شروع می‌کنیم.

۲. شاخه‌ی context را با bilinear up sample به همان رزولوشن شاخه‌ی detail می‌رسانیم.

۳. روی هر دوی آن‌ها یک pointwise Conv2D ۱×۱ می‌زنیم تا تعداد channel ها هماهنگ شود.

۴. سپس به‌سادگی دو تنسور را با هم add می‌کنیم.

۵. بلافاصله بعد ازadd ، یک تابع غیرخطی (f = ReLU) اعمال می‌شود.

۶. در نهایت یک depthwise convolution (DWConv) برای smooth کردن فیچرها و حذف آرتیفکت های احتمالی اجرا می‌شود.

این طراحی خیلی ساده‌تر و سبک‌تر از روش‌های پیچیده‌ی fusion مثلconv + concatenation های سنگین یا attention است و به‌خوبی بین accuracy و speed تعادل برقرار می‌کن.

3.2.4 Classifier

In the classifier we employ two depthwise separable convolutions (DSConv) and one pointwise convolution (Conv2D). We found that adding few layers after the feature fusion module boosts the accuracy. The details of the classifier module is shown in the Table 1.

Softmax is used during training, since gradient decent is employed. During inference we may substitute costly softmax computations with argmax, since both functions are monotonically increasing. We denote this option as Fast-SCNN cls (classification). On the other hand, if a standard DCNN based probabilistic model is desired, softmax is used, denoted as Fast-SCNN prob (probability).

تصویر ۴ - بخش **classifier** مقاله مرجع

در ماژول Classifier(بخش ۳.۲.۴) دقیقاً این اتفاق می‌افته:

۱. بعد از Feature Fusion ، از دو تا depthwise separable convolution (DSConv) و سپس یک pointwise convolution (Conv2D) استفاده می‌کنیم.

۲. نویسنده‌ها می‌گویند که adding few layers after the feature fusion module boosts the accuracy، پس همین چند لایه‌ی ساده باعث افزایش دقت می‌شود.

۳. Softmax در زمان training به‌کار میره تا Gradian decent اعمال بشود.

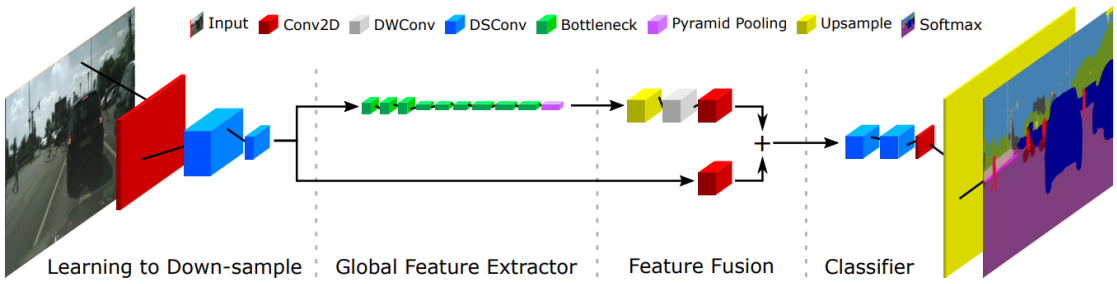
۴. برای inference ، به‌جای Softmax می‌شه از argmax استفاده کرد چون هر دو monotonic هستن و سرعت رو بالا برد این حالت رو Fast-SCNN cls می‌گویند.

۵. اگر خروجی probability بخوایم ، Softmax رو نگه می‌دارید و مدل رو Fast-SCNN prob می‌گویند.

Input	Block	Expansion	channels	n	stride
1024×2048×3	Conv2D	–	32	1	2
512×1024×32	DSConv	–	48	1	2
256×512×48	DSConv	–	64	1	2
128×256×64	bottleneck	6	64	3	2
64×128×64	bottleneck	6	96	3	2
32×64×96	bottleneck	6	128	3	1
32×64×128	PPM	–	128	–	–
32×64×128	FFM	–	128	–	–
128×256×128	DSConv	2	128	2	1
128×256×128	Conv2D	–	19	1	1

جدول ۱ - خلاصه لایه ها

ساختار مدل



تصویر ۵- ساختار مدل **Fass-CNN**

اشتراک محاسبات

لایه‌های اولیه بین دو مسیر به‌جای محاسبات تکراری در دو شاخه باعث بهبود کارآمدی شده است.

در مقایسه با U-Net و سایر انکدر دیکدرها، Fast-SCNN بسیار سبک‌تر و سریع‌تر است، آن هم بدون کاهش چشمگیر در دقت روی تصاویر شهری با رزولوشن بالا.

در مقایسه کامل Fast-SCNN با معماری‌های کلاسیک Encoder–Decoder مثل U-Net وFCN ، چند جنبه کلیدی داریم .

۱. ساختار معماری(Architecture)

در مدل‌های Encoder–Decoder مانند U-Net، معماری به دو بخش مجزا تقسیم می‌شود:

- Encoder که تصویر ورودی را مرحله به مرحله downsample می‌کند و ویژگی‌های سطح بالا (semantic) را استخراج می‌کند.
- Decoder که این ویژگی‌ها را با کمک مجموعه‌ای ازskip-connection ها در رزولوشن‌های مختلف با مقادیر اصلی تصویر تا حد ممکن بازسازی می‌کند و یک segmentation map دقیق می‌سازد.

به‌عکس، Fast-SCNN با ایده multi-branch عمل می‌کند:

۱. یک detail path برای استخراج ویژگی‌های low-level (لبه‌ها، بافت‌های ساده) در رزولوشن نسبتاً بالا.

۲. یک context path برای گرفتن global context در رزولوشن پایین‌تر از طریق efficient Pyramid Pooling bottleneck blocks

۳. فقط یک skip connection ساده بین ابتدای detail path و ابتدای context path قرار می‌گیرد تا محاسبات اولیه بین هر دو شاخه shared باشد.

۴. در نهایت، Feature Fusion Module همه اطلاعات را با هم ترکیب می‌کند.

این طراحی اجازه می‌دهد که به‌جای دو شبکه سنگین Encoder و Decoder با چندین اتصال، یک ساختار سبک و سریع ولی همچنان دقیق داشته باشیم.

۲. حجم پارامترها(Model Size)

- U-Net / FCN
  مدل‌های استاندارد معمولاً بین ۲۰ تا ۳۰ میلیون پارامتر دارند (یا حتی بیشتر)، چون هم Encoder و هم Decoder لایه‌های متعددی از Conv2D و skip connection دارند.
- Fast-SCNN:
  تنها حدود 1.11میلیون پارامتر! این کاهش عظیم با دو تکنیک اصلی حاصل می‌شود:

  استفاده از depthwise separable convolution (DSConv) در همه جا به‌جز لایه اول.

به‌کارگیری inverted residual blocks شبیه MobileNet-V2 در بخشcontext ، که با expansion و depthwise convolution پارامترها را به حداقل می‌رساند.

۳. هزینه محاسباتی و سرعت(Computational Cost & Speed)

- U-Net / FCN:

به‌دلیل محاسبات سنگین در Decoder upsampling و convolutions متعدد و skip connection های مکرر، روی تصاویر با رزولوشن بالا معمولاً قادر به real-time نیستند.

- Fast-SCNN:

طراحی شده برای real-timeیا حتی فوق:real-time

  - سرعت تخمینی 123.5 fps روی تصاویر ۲۰۴۸×۱۰۲۴
  - کاهش FLOPs به دلیل DSConv وinverted residual
  - Feature Fusion ساده با یک add و یک DWConv

۴. حفظ جزئیات مرزی(Boundary Preservation)

- U-Net

skip-connection های متعدد اجازه می‌دهند که جزئیات مرزی (لبه‌ها و بافت‌های ریز) بدون افت کیفیت به Decoder منتقل و بازسازی شود. بنابراین خروجی‌ها بسیار دقیق هستند.

- Fast-SCNN

از آنجا که فقط یک اتصال ساده در ابتدای مسیر وجود دارد، حفظ جزئیات مرزی در حد U-Net نیست؛ اما:

  - شاخه detail path در رزولوشن بالا ویژگی‌های لبه را خوب استخراج می‌کند.
  - Feature Fusion با یک DWConv در انتها کمک می‌کند آرتیفکت ها کاهش یابند و مرزها نسبتا تیز نگه داشته شوند.

در عمل، دقت مرزی Fast-SCNN تا حدود زیادی مناسب است، به‌ویژه وقتی هدف یک سیستم-real timeباشد.

۵. نیاز به پیش‌آموزش(Pre-training)

- U-Net / FCN

معمولاً نیازمند pre-training روی مجموعه‌های بزرگ مانند ImageNet هستند تا استخراج ویژگی‌های سطح بالا به خوبی انجام شود.

- Fast-SCNN

با ظرفیت کوچک‌تر و طراحی بهینه، می‌تواند از صفر (scratch) و تنها با Data Augmentation آموزش داده شود و باز هم به دقت مطلوب برسد. این یعنی وابستگی کمتری به مدل‌های از پیش آموزش‌دیده دارید.

۶. کاربرد در دستگاه‌های تعبیه‌شده(Embedded / Mobile)

- **U-Net / FCN**

حجم بالا و FLOPs زیاد باعث می‌شود اجرای آن‌ها روی دستگاه‌هایی مثل موبایل یا ربات‌های سبک عملاً غیرممکن یا خیلی کند باشد.

- Fast-SCNN

به‌خاطر:

  - کم بودن پارامترها(1.11M) ،
  - استفاده از DSConv وinverted residual ،
  - و طراحی سادهFeature Fusion ،بسیار مناسب پیاده‌سازی روی سخت افزارهای با حافظه و قدرت محاسباتی محدود است.

اگر نیازمند این بودیم که در لحظه (real-time)روی تصاویر با رزولوشن بالا سگمنتیشن انجام دهیم و در عین حال مدلمان روی دستگاه‌های سبک و تعبیه‌شده قابل اجرا باشد، Fast-SCNN با ساختار-multi branchو ماژول‌های بهینه‌شده DSConv ، inverted residual، PPMو Feature Fusion ساده یک انتخاب بسیار مناسب است. اما اگر دقت حداکثری و حفظ جزئیات ریز اولویت دارد و منابع محاسباتی نامحدود در اختیار داریم، معماری‌های کلاسیک Encoder–Decoder باskip-connection های متعدد مثل U-Net هنوز قدرت بیشتری در بازسازی دقیق مرزها و بافت‌ها دارند.

۱-۲ آماده سازی مجموعه داده

در این بخش، مجموعه داده‌ی CamVid را برای آموزش مدل آماده کردیم. وظایف اصلی عبارت بودند از:

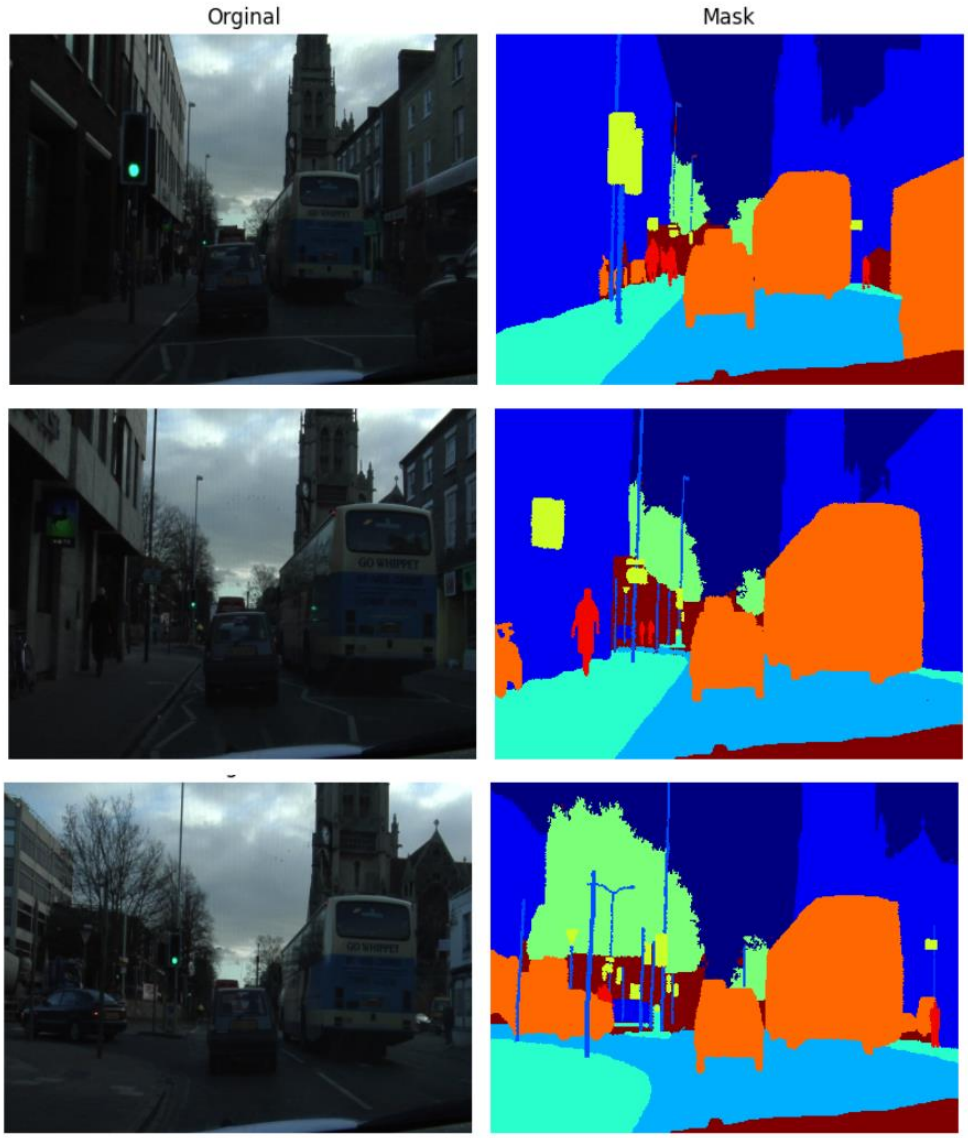
۱. دانلود و سازمان‌دهی پوشه‌های تصویری و ماسک انجام شد.
۲. استخراج لیست فایل‌هایTrain/Validation/Test انجام شد.
۳. شمارش تعداد نمونه‌ها در هر زیرمجموعه به درستی انجام شد.
۴. چند نمونه‌ی ورودی به همراه ماسک متناظر نمایش داده شد.

تعداد نمونه به شرح زیر هستند:

تعداد نمونه های train : ۳۶۷ عدد

تعداد نمونه های validation : ۱۰۱ عدد

تعداد نمونه های test : ۲۳۳ عدد



تصویر ۶ - چند نمونه از تصاویر اصلی به همراه با ماسک

۳-۱ بهینه ساز، متریک ها و تابع هزینه

در مسائل دسته‌بندی (Classification) و به‌ویژه در مسائل تفکیک‌پذیری تصویر Image Segmentation، معیارهای متداول مانند دقت (Accuracy) به‌تنهایی نمی‌توانند عملکرد واقعی مدل را به خوبی منعکس کنند؛ چرا که میزان نابرابری در برچسب‌های مثبت و منفی (مثلاً پیکسل های پس‌زمینه و پیش زمینه) می‌تواند باعث فریب خوری شود. دو متریک Dice Coefficient و Intersection over Union (IoU)به دلیل حساسیت بالاتر به توازن صحیح بین پیش‌بینی و واقعیت، به‌طور گسترده تری در این حوزه به‌کار می‌روند.

چه متریک‌هایی برای ارزیابی segmentation استفاده کردیم؟

۱- Dice Coefficien

$$\text{Dice}(P,G) = \frac{2|P \cap G|}{|P| + |G|}$$

فرمول ۱- **Dice**

که در آن Pمجموعه پیکسل‌های پیش‌بینی شده به‌عنوان پیش‌زمینه و Gمجموعه پیکسل‌های درست (Ground Truth)است.

مقدار Dice بین ۰ و ۱ قرار دارد. عدد نزدیک به ۱ نشان‌دهنده همپوشانی بالاست.

۲- Intersection over Union (IoU):

می‌خواهیم بدانیم چقدر «همپوشانی» بین ناحیه‌ای که مدل برای هر کلاس پیش‌بینی کرده و ناحیه واقعی همان کلاس وجود دارد IoU .معیاری بین ۰ و ۱ است که عدد بزرگ‌تر بهتر است.

$$\frac{TP_c}{TP_c + FP_c + FN} = \frac{|\text{Pred}_c \cap \text{GT}|}{|\text{Pred}_c \cup \text{GT}|} = \text{IoU}_c$$

فرمول ۲ – **IoU**

IoUنیز بین ۰ و ۱ است و مشابه Dice همپوشانی پیش‌بینی و واقعیت را می‌سنجد، اما وزن دهی کمتری به «دو برابر کردن اشتراک» می‌دهد.

True Positive (TP):تعداد پیکسل‌هایی که هم مدل برای کلاس c زده و هم واقعاً در Ground Truth کلاس c هستند.

False Positive (FP): تعداد پیکسل‌هایی که مدل برای c زده ولی اشتباه است.

False Negative (FN): تعداد پیکسل‌هایی که مدل برای c نزده ولی در حقیقت c هستند.

این دو متریک به صورت دستی پیاده سایزی شدند و در آموزش و ارزیابی مدل مورد استفاده قرار گرفتند.

۴-۱ پیاده سازی مدل

در این بخش، مدل **Fast-SCNN** را مطابق مقاله برای مسئله سگمنتیشن شهری CamVid با ۱۱ کلاس پیاده‌سازی و تعداد کل پارامترها را گزارش کرده‌ایم. سپس ساختار سه بلاک اصلی را به‌صورت مختصر توضیح می‌دهیم.

خلاصه پیاده‌سازی

زیرشبکه LearningToDownsample:

یک Conv معمولی ۳→۳۲، stride=2

یک بلوک DSConv ۳۲→۴۸، stride=2

یک بلوک DSConv ۴۸→۶۴، stride=2

GlobalFeatureExtractor:

پنج InvertedResidual با ابعاد (۶۴→۶۴)، (۶۴→۶۴)، (۶۴→۹۶)، (۹۶→۹۶(،)۹۶→۱۲۸(

PyramidPoolingروی خروجی ۱۲۸ کاناله

کاهش کانال به ۱۲۸ با1×Conv1

FeatureFusion:

ترکیب فیچرهای رزولوشن بالا (۶۴ کاناله) و خروجی GFE (۱۲۸ کاناله) با upsample و دو بلوک DSConv.و ۱×۱

Classifier:

یک DSConv ۱۲۸→۱۲۸ (و یک) Conv1×1 ۱۲۸→۱۱ و در نهایت upsample تا ابعاد اصلی تصویر.

تعداد پارامترها:

مدل حدود ۱۱۳۵۸۳۵ پارامتر دارد که برای یک معماری real-time فوق‌العاده سبک است.

توضیح مؤلفه‌های کلیدی:

Depthwise Separable Convolution

- **Depthwise Convolution**:

- یک کرنل k * k مستقل روی **هر کانال** ورودی اجرا می‌شود .
- تعداد پارامترها: k * k * in_ch

- **Pointwise Convolution**:
  - یک 1*1 Conv برای ترکیب کانال‌ها و تبدیل از in_ch به out_ch.
  - تعداد پارامترها: out_ch * in_ch

- **مزیت** :کاهش چشمگیر پارامتر و محاسبات نسبت به Conv ساده که $k^2 * out\_ch * in\_ch$ پارامتر دارد.
- **کاربرد** :در همه بلوک‌های DSConv مدل برای حفظ سرعت و سبکی.

۳-۲. Inverted Residual Block

- **Expand**با Conv1×1: افزایش کانال داخلی به t * in_ch
- **Depthwise**با Conv3×3 روی کانال‌های گسترش‌یافته.
- **Project**با Conv1×1: کاهش کانال به out_ch.
- **Residual Connection**وقتی که stride=1و in_ch=out_chبرقرار است.
- **ایده** :فضای ویژگی را ابتدا «باز» کرده، سپس پردازش عمیق‌تری انجام داده و در نهایت «فشرده» می‌کند.

۳-۳. Pyramid Pooling Module

- چهار شاخه AdaptiveAvgPool2d با اندازه‌های {1,2,3,6}
- خروجی هر شاخه را با Conv1×1 به in_ch/4 کانال کاهش می‌دهیم.
- سپس هر خروجی را upsample و با خروجی اصلی Concate می‌کنیم تا ویژگی‌های چندمقیاسی در دسترس باشند.

با این پیاده‌سازی و استفاده از بلوک‌های Depthwise Separable Convolution، Inverted Residualو Pyramid Pooling، مدل Fast-SCNN ضمن حفظ دقت مناسب، به‌طرز چشمگیری سبک و سریع باقی می‌ماند.

۱-۵ آموزش مدل

در این مرحله، مدل Fast-SCNN را با تنظیمات دلخواه خود آموزش داده و روند یادگیری را از طریق نمودارهایLoss ، mIoUو Dice روی داده‌های آموزش و اعتبارسنجی دنبال کردیم. همچنین پارامترهای مهم شاملbatch size ، epochs، learning rate، بهینه‌ساز و در صورت لزوم scheduler را مشخص کردیم

هدف ما آموزش یک شبکه سبک و سریع به نام Fast-SCNNبرای سگمنتیشن صحنه‌های شهری روی دیتاست CamVidبا ۱۱ کلاس است. کل فرایند از آماده‌سازی داده شروع شد، سپس ساخت دیتاست و ترنسفورم‌ها، پیاده‌سازی مدل، تعریف تابع هزینه و بهینه‌ساز، و در نهایت حلقه آموزش همراه با ارزیابی و ذخیره بهترین مدل انجام شد. در هر مرحله با چالش‌هایی روبه‌رو شدیم که برایتان شرح می‌دهم.

۱- در ابتدا داده ها همان طور که بالا توضیح داده شده آماده شدند و گزارش های مربوطه انجام شد. در طی این فرایند به چند ماسک رنگی که به صورت خراب ذخیره شده بود برخوردیم که آن هارا حذف کردیم.

۲- در مرحله بعد شروع به ساخت Data Set و ترنسفورم ها کردیم.

نکته ای که باید به آن توجه میکردیم برای agumentaion این بود که اگر RandomResizedCropیا Rotateرا جداگانه روی تصویر و سپس روی ماسک اعمال می‌کردیم، گاهی ابعاد ماسک و تصویر متفاوت می‌شد.

برای حل این مشکل ،یک کلاس JointTransformکه تمام عملیاتcrop ، rotate، flipو color-jitter را هم‌زمان روی (img, mask) اجرا کند نوشته شد.

۳- در مرحله بعد مدل را مطابق با مقاله طراحی کردیم که جزییات در صفحات بالا موجود است.

۴- برای آموزش مدل از متریک هایی استفاده کردیم که به شرح زیر هستند.

- **Pixel Accuracy**
- **mIoU**
- **Dice Coefficient**

توابع هزینه نیز :

CrossEntropyLoss(weight=class_weights, ignore_index=255)

class_weightsرا با محاسبه فراوانی هر کلاس در مجموعه Train و معکوس‌سازی آن تنظیم کردیم.

label smoothing / focal loss:

برای محاسبه مقدار loss از این روش استفاده کردیم تا مدل بهتر و راحت تر آموزش ببیند.

ابتدا label_smoothing=0.1 را امتحان کردیم؛ خیلی تفاوت خاصی نداشت.

سپس یک پیاده‌سازی ساده FocalLoss (γ=2) نوشتیم و ترکیبی از focal + CE استفاده کردیم؛ *چالش :*کمی آموزش ناپایدار شد و باید α و γ را دقیق‌تر تنظیم می‌کردیم.

۵- تنظیمات بهینه‌سازی

Optimizer:

AdamW(lr=1e-3, weight_decay=1e-5)

به دلیل جداسازی weight decay از گرادیان.

Scheduler:

OneCycleLR(max_lr=5e-4, pct_start=0.3, div_factor=10, final_div_factor=100)

چالش :انتخاب div_factor و pct_start کمی آزمون و خطا نیاز داشت تا نرخ یادگیری اوج مناسبی پیدا کند.

Gradient Clipping:

clip_grad_norm_(…, max_norm=1.0)

برای جلوگیری از نوسان شدید گرادیان‌ها در بالای شبکه.

Early-Stopping:

نظارت بر mIoU بر داده های validation اگر ۱۰ epoch متوالی بهبود نداشت، آموزش متوقف شود.

۶- حلقهٔ آموزش و نتایج

در آموزش نوسانات ناگهانی ValLoss داشتیم؛ با کاهش learning rate اولیه از 1e-3به 5e-4 افت شدیدی در نوسان‌ها ایجاد شد اما overall کندتر همگرا شد، پس مجدد 1e-3 با scheduler انتخاب شد.

اوج متریک‌ها

Val mIoU ≈0.5489

Val Dice ≈0.6463

Early-Stopping

در epoch ۹۴ به دلیل عدم بهبود متوقف شد.

ارزیابی روی تست

mIoU = 0.3975

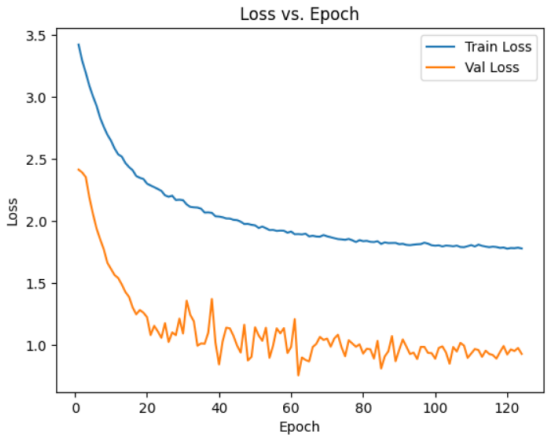
Dice =0.4952

Accuracy =0.7779

توضیح: افت ~۰.۱۵ در mIoU از valid تا تست نشان‌دهنده چالش‌های Generalization و اختلاف چالش‌برانگیزی صحنه‌هاست.

با پیمودن این مراحل و حل تدریجی چالش‌ها، به یک **pipeline** کامل رسیدیم که:

- داده‌ها را به صورت یکپارچه بارگذاری و augment می‌کند،
- مدل Fast-SCNN سبک را با ۱.۱ میلیون پارامتر اجرا می‌کند،
- از متریک‌های استاندارد mIoU و Dice برای انتخاب بهترین مدل استفاده می‌کند،
- و در نهایت عملکرد معقولی (Dice>0.5) روی valid و تست ارائه می‌کند.



نمودار ۱ - در نمودار **«Loss vs. Epoch»**

در نمودار «Loss vs. Epoch» می‌بینیم:

۱. کاهش سریع در دهه اول

- مقدار Train Loss از حدود ۳.۴ در epoch اول به حدود ۲.۳ تا پایان epoch 10 سقوط می‌کند.
- این نشان می‌دهد که مدل در دهه اول یادگیریِ ویژگی‌های پایه تصویر (مثل لبه‌ها و بافت‌های ساده) خوب عمل کرده است.

۲. کاهش پایدار ولی آهسته‌تر پس از آن

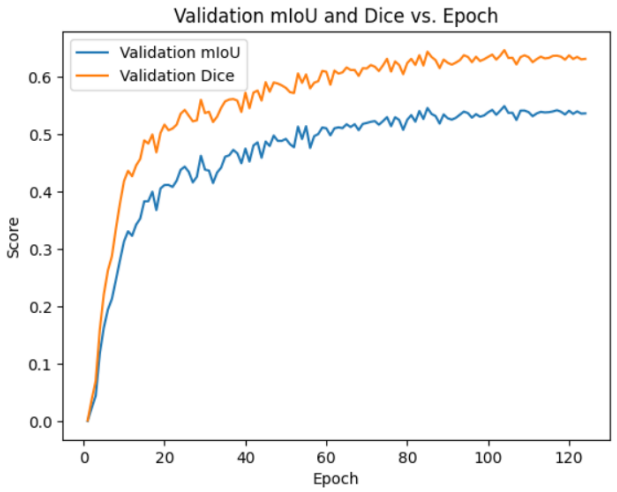
- از epoch 10 تا ۸۰، Train Lossبه آرامی از ۲.۳~ به ۱.۹~ کاهش می‌یابد.
- این بخش از منحنی معمولاً زمانی است که مدل باید جزئیات پیچیده‌تر و مرزهای ظریف کلاس‌ها را فرا بگیرد؛ طبیعتاً یادگیری سخت‌تر و تدریجی‌تر است.

۳. رفتار Validation Loss

- Val Loss از ۲.۴ در ابتدای آموزش به ۱.۱تا پایان epoch 20 کاهش می‌یابد.
- پس از epoch 20 ، Val Lossبین ۱.۰ و ۱.۲ نوسان می‌کند و در نهایت کمی حول ۱.۰ تثبیت می‌شود.
- نوسانات کوچک در این بخش معمول است و نشان‌دهنده‌ی sensitivity مدل به مثال‌های معتبر و پیچیده‌ی والید است.

۴. فاصله Train و Val Loss

- همیشه Val Loss کمتر از Train Loss است به‌دلیل وزن‌دهی و regularization ، اما فاصله ثابت بین آن‌ها نشان می‌دهد overfitting گسترده‌ای رخ نداده است.
- این فاصله پایدار، در ترکیب با Early-Stopping بر اساس mIoU ، تضمین می‌کند مدل بیش‌ازحد داده‌های والید را هم یاد نگیرد.



نمودار ۲-نمودار «**Validation mIoU** و **Dice vs. Epoch**»

۱. رشد سریع در دهه اول

- تا حدود epoch 10 هر دو متریک به‌شدت صعود می‌کنند:
  - mIoU از تقریباً صفر به ۰.۳۵
  - Dice از صفر به ۰.۵
- این رشد سریع نشان می‌دهد که مدل در دهه اول دارد ویژگی‌های کلیِ صحنه (جاده، آسمان، ساختمان) را به‌خوبی یاد می‌گیرد.

۲. پیشی گرفتن Dice از mIoU

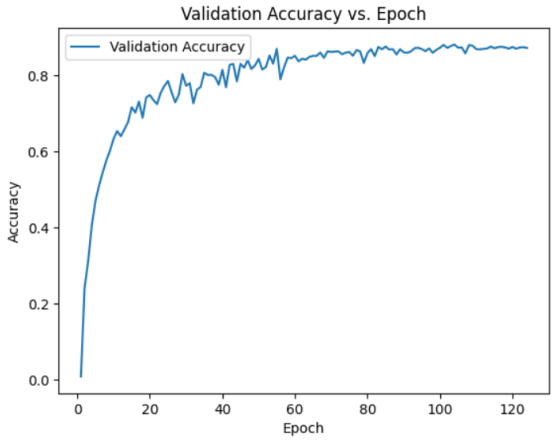
- به محض اینکه Dice از ۰.۴ عبور می‌کند، mIoU پشت سرش حرکت می‌کند.
- چون Dice به اشتراک پیکسل‌های درست بیشتر وزن می‌دهد، سریع‌تر بالا می‌رود و نسبت به mIoU نرم‌تر است.

۳. نوسان های نمودار ها

- بعد از حدود epoch 30 هر دو منحنی آرام شد و:
  - mIoU حول ۰.۴۸–۰.۵۲ نوسان می‌کند
  - Dice حول ۰.۵۵–۰.۶۲ باقی می‌ماند
- این مرحله معمولاً وقتی است که مدل به بخش‌های ظریف‌تر (مرزهای باریک، اشیاء کوچک) می‌رسد و پیشرفت کند می‌شود.

۴. نوسانات جزئی در نواحی بالاتر

- بین epoch 30 تا ۸۰، گاهی Dice به اوج ~۰.۶۳ می‌رسد و گاهی اندکی افت می‌کند.
- این نوسانات طبیعی است و دلایل احتمالی آن عبارت‌اند از:
  - Sensitivity مدل به نمونه‌های «سخت» در والید
  - تغییرات کوچک در یادگیری که Scheduler و Early-Stopping مدیریت می‌کنند.



نمودار ۳-نمودار **«Validation Accuracy vs. Epoch»**

۱- **رشد سریعِ اولیه(Epoch 1–10)**

- از صفر شروع می‌کند و در عرض کمتر از ۱۰ epoch به حدود ۰.۶–۰.۷ می‌رسد.
- این افزایش سریع معمولا به خاطر یادگیری آسان‌ترین ویژگی‌ها (جاده، آسمان، ساختمان) است که بخش عمده پیکسل‌ها را تشکیل می‌دهند.

۲- **نوسانات کوچک و تثبیت(Epoch 10–30)**

- بین epoch 10 و ۳۰، نمودار Accuracy حول ۰.۷–۰.۸ نوسان می‌کند و به تدریج به سمت مقدار بالاتر حرکت می‌کند.
- این مرحله نشان می‌دهد که مدل در یادگیری کلاس‌های میانی و مرزهای ساده‌تر پیشرفت می‌کند.

۳- **در مقادیر بالا(Epoch 30–80)**

- پس از epoch 30 Accuracy به حدود ۰.۸–۰.۸۷ صعود کرده و سپس حول این ناحیه تثبیت می‌شود.
- نوسانات جزئی (spikes) در این محدوده طبیعی است و می‌تواند ناشی از مثال‌های سخت در مجموعه valid باشد.

Accuracy بالا ≠ کیفیت بالای segmentation :

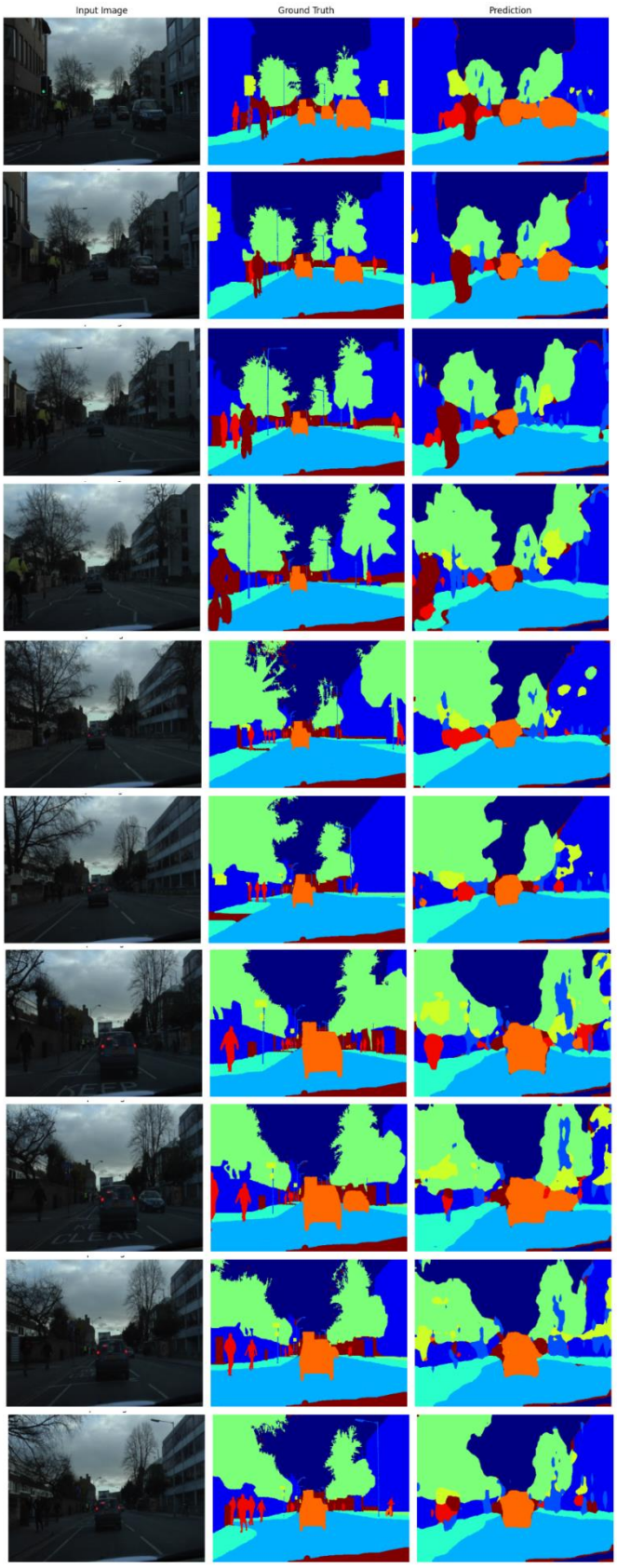
چون در CamVid پیکسل‌های برخی کلاس‌ها (مثل جاده و آسمان) بسیار زیادند، مدل با تمرکز روی این کلاس‌ها می‌تواند Accuracy بالایی داشته باشد ولی مرزها یا اشیاء کوچک را درست تشخیص ندهد. به همین خاطر حتی وقتی Accuracy ≈ 0.85 است، mIoU و Dice تنها ۰.۵–۰.۶ می‌شوند.

اهمیت نگاه چندمتریکی:

برای ارزیابی کامل، باید همیشه به همراه Accuracy به mIoU و Dice هم دقت کنیم تا مطمئن شویم مدل کلاس‌های کوچک‌تر و مرزها را هم یاد می‌گیرد.

۱-۵ ارزیابی مدل

پس از پایان آموزش، ۱۰ تصویر تصادفی از مجموعه تست را همراه با ماسک واقعی و ماسک پیش‌بینی‌شده مدل Fast-SCNN نمایش دادیم. در زیر یکی از این نمونه‌ها را می‌بینید



تصویر ۷-۱۰ تصویر تصادفی از مجموعه تست

کلاس‌های غالب (آسمان، جاده، ساختمان)

- مدل در تشخیص آسمان (رنگ آبی تیره) و جاده (آبی روشن) عملکرد مطلوبی دارد؛ مرز این نواحی معمولاً واضح است و تقریباً همپوشانی بالایی با ماسک واقعی دارد.

درختان و پوشش گیاهی

- ناحیه درختان (سبز روشن) با اینکه بافت پیچیده‌ای دارد، در اغلب تصاویر به خوبی تفکیک می‌شود؛ ولی در مرزهای ظریفِ شاخ و برگ ممکن است نویزهای ریز دیده شود.

وسیله‌های نقلیه و عابران

- خودروها (بنفش/نارنجی) در ابعاد متوسط و بزرگ خوب شناسایی می‌شوند؛ اما در مواقعی که چند خودرو به هم نزدیک‌اند یا بخش‌هایی از آن‌ها توسط اجسام دیگر پوشیده شده، پیش‌بینی به هم‌ریخته و بخش‌هایی از ماشین یا عابر را از دست می‌دهد.
- دوچرخه‌سوار و عابر پیاده (قهوه ای /قرمز) که در تصویر اول کنار خیابان قرار دارند، عمدتاً شناسایی شده‌اند اما پیکسل‌هایی از زمینه (جاده) به آن‌ها نسبت داده شده است.

اشیاء کم‌نمونه (تیربرق، علائم راهنمایی)

- علامت‌های راهنمایی و تیرهای برق (رنگ‌های زرد) اغلب اشتباهاً به کلاس مجاور (مثلاً درخت یا خودرو) تخصیص داده می‌شوند؛ این موضوع ناشی از تعداد کم نمونه‌های آموزشی و تقارن با پس‌زمینه است.

مرزهای پیچیده و سایه‌ها

- در لبه‌ها و نواحی سایه‌دار (مثل کناره خیابان)، مدل گاهی دیدگاه روشنی ندارد و پیکسل‌های Voidیا Background را به کلاس‌های دیگر اضافه می‌کند.

و در انتهای فرآیند، عملکرد نهایی مدل بر روی کل مجموعه تست به صورت زیر گزارش شد:

mIoU = 0.3975

متوسط همپوشانی بین پیش‌بینی و ماسک واقعی حدود ۴۰٪ است. این مقدار برای یک مدل سبک و real-time مانند Fast-SCNN معقول بوده اما جای پیشرفت دارد.

Dice = 0.4952

مقدار Dice نزدیک به ۰.۵ نشان می‌دهد که تقریباً در نصف نواحی مدل و Ground Truth هم‌پوشانی مناسب داریم. Dice نسبت به mIoU نرم‌تر است و این عدد را بالا برده است

Accuracy = 0.7779

تقریباً ۷۸٪ از پیکسل‌ها درست دسته‌بندی شده‌اند. این عدد بالاتر از mIoU است چرا که بسیاری از پیکسل‌ها (مثلا جاده و آسمان) به راحتی تفکیک می‌شوند و کلاس‌های غالب را تشکیل می‌دهند.

پرسش۲ – پیاده سازی یک سیستم طبقه بندی خودرو با استفاده از VGG16 و SVM

۲-۱ مقدمه