**Note :This file will be complete by Moheb and is available in his GitHub as below link :**

[**https://github.com/MohebZandi/Comupter\_Vision/blob/main/A%20Deep%20Understanding%20of%20Deep%20Learning.docx**](https://github.com/MohebZandi/Comupter_Vision/blob/main/A%20Deep%20Understanding%20of%20Deep%20Learning.docx)

**A Deep Understanding of Deep Learning**

**by Mike X Cohen**

فولدر 3

* در مدل های دیپ لرنینگ از ریاضیات ساده در حد دبیرستان یعنی استفاده از چهارعمل اصلی و تعدادی از توابع ساده مثل مثلثات یا لگاریتم استفاده می شود
* در لایه های شبکه از این توابع و عملیات ریاضی به تعداد بسیار زیاد استفاده می شود و همین باعث امکان انجام محاسبات پیچیده می گردد
* مدل کلی شبکه های عصبی عمیق به این صورت است که ورودی را گرفته و خروجی را پیش بینی می نماید
* برای پیش بینی در کاربردهای مختلف، استفاده از شبکه های عصبی متفاوتی توصیه شده است. مثلا برای پردازش و پیش بینی تصاویر از سی ان ان استفاده می شود و برای پردازش متن یا ترجمه ویا تحلیل سری های زمانی از آر ان ان. برای انجام محاسبات دسته بندی و خوشه بندی از ANN استفاده می شود
* نکته مهم این است که ساختار همه شبکه های عمیق مشابه هم است... مانند ساختمان ها که ممکن است از نظر کاربری و نما و امکانات متفاوت باشند ولی زیر ساخت همه آنها مشابه هم است و از آجر و سیمان و لوله و سیم وغیره تهیه شده اند.
* حرکت رو به جلو به معنی اختصاص وزن های پارامتر های ورودی به معادله می باشد و حرکت برگشت به معنی اخذ بازخورد و نیاز به اصلاح وزن پارامتر ها می باشد. Forward Propagation و Backward Propagation
* مثال حرکت رو به جلو و برگشت: تهیه ساندویچ کره و مربا : تعیین میزان کره و مربا در حرکت رو به جلو و اخذ بازخورد از مشتری در میزان شیرین یا چرب بودن ساندویچ و اصلاح میزان استفاده از آنها در حرکت رو به جلو بعدی
* مشابهت شبکه های عصبی با سلول های عصبی انسان، قیاس درستی نیست و مثل تشابه خودرو و اسب و یا هواپیما با پرنده است. تکنولوژی خصوصیات خود را دارد و قیاس آن با طبیعت درست نیست.

فولدر 5

* تفاوت Complicated , Complex :
  + اشیاء Complicated شامل تعداد زیادی از اجزا ساده هستند که در کنار یکدیگر کار کرده و مجموعه پیچیده ای را بوجود می آورند مثل خودرو یا گوشی موبایل
  + اشیاء Complex دارای تعداد اجزا کمی هستند ولی پیچیدگی آنها زیاد است. مثل علوم پزشکی و Conway’s game of Life
* تفاوت Dummy variables و One-hot encoding :
  + متغیرهای دامی در واقع بصورت صفر و یک وجود یا عدم وجود مورد خاصی را برای یک متغیر مشخص می کنند. مثلا شخص در امتحان قبول یا رد شده است. خروجی یک بردار یا ماتریس یک بعدی است.
  + در one-hot encoding چند متغیر داریم که می خواهیم وجود یا عدم وجود آن را بررسی کنیم. در واقع از چند متغیر دامی تشکیل می شود. مثلا شخص در درس ریاضی، تاریخ و علوم نمره قبول یا رد دارد. خروجی یک ماتریس چند بعدی است که از چند ماتریس یک بعدی (دامی) تشکیل شده است.
* تعریف تابع سافت مکس (Softmax):
  + این تابع با محاسبه احتمال هر یک از اعداد ورودی، مجموعه ای از احتمال ها را بعنوان خروجی بر می گرداند. جمع خروجی ها برابر یک می باشد.
  + فرمول محاسبه احتمال برای هر i برابر است با :
  + بزرگترین عدد در این تابع بیشترین مقدار احتمال و کوچکترین آنها، کمترین احتمال را دارد.
* آنتروپی و کراس آنتروپی
  + به‌طور کلی در علوم و مهندسی، آنتروپی معیاری از میزان ابهام یا بی‌نظمی است.
  + آنتروپی به معنای این است که هر چه میزان عدم قطعیت یا پیش بینی پذیر بودن مقادیر بالا تر یا پایین تر باشد، مقدار آنتروپی به صفر نزدیکتر می شود. یعنی هرچه احتمال رخداد اتفاقی بیشتر باشد، آنتروپی آن کمتر است. از طرفی هر چقدر احتمال رخ دادن اتفاقی کمتر باشد، آنتروپی آن کمتر است. مثلا در پرتاب سکه، احتمال آنکه از 100 پرتاب هر صد مرتبه شیر بیاید نزدیک به صفر است و آنتروپی آن هم صفر است. آنتروپی احتمال آنکه 50 درصد شیر بیاید یک است. هر چقدر احتمال شیر آمدن سکه به ۰٫۵ نزدیکتر باشد ابهام در مورد نتیجه آن بیشتر است و اطلاع از نتیجه، به‌طور میانگین اطلاعات بیشتری دربردارد.

* + برای شرایطی که دو خروجی متصور هستیم مثل تشخیص تصویر گربه که در واقع خروجی درست یا غلط هست فرمول آنتروپی به شرح زیر است:

اگر احتمال p= 0.25 باشد، آنگاه آنتروپی یعنی ابهام در تشخیص نتیجه حدود 0.56 می باشد.

* + کراس آنتروپی احتمال رخ داد یک اتفاق به شرط اتفاق دیگر را بررسی می کند. در واقع در این زمینه دوتوزیع احتمال دخیل هستند.

* + در مثال تشخیص گربه اگر دسته بندی گربه بصورت صفر و یک باشد و احتمال دیده شدن گربه 0.25 و دیده نشدن 0.75 باشد، آنگاه فرمول محاسبه کراس آنتروپی باینری (صفر و یک) به شرح زیر است.

* گرادیان کاهشی (Gradient Descent) : برای پیدا کردن نقطه مینیموم تابع می توانیم از مشتق تابع استفاده کنیم و در نقطه ای که شیب (مشتق) برابر صفر می شود (نقطه ماکزیموم یا مینیموم) با توجه به مقدار مشتق در قبل و بعد از نقطه مشخص شده تشخیص دهیم که مینیموم یا ماکزیموم است. در نقطه مینیموم مقدار مشتق یا شیب خط قبل از نقطه منفی و بعد از آن مثبت می شود.
* گرادیان محو شونده (Vanishing Gradient) : در مواقعی ممکن است مقدار شیب خط یا مشتق تابع بدون اینکه در نقطه مینیموم یا ماکزیموم باشد برابر با صفر شود. این نقاط جایی هستند که مقدار تابع ثابت (Constant) می شود و شیب یا مشتق آن تغییر نمی کند و برابر صفر می شود. به این حالت گرادیان محو شونده (Vanish) گویند.

فولدر 6 : گرادیان کاهشی

یکی از مهمترین الگوریتم های شبکه های عصبی و یادگیری عمیق الگوریتم گرادیان کاهشی است و بدون آن بی معنی هستند

* چگونگی یادگیری عمیق توسط مدل:
  + تخمین پارامترها
  + محاسبه خطا از پارامترهای تخمینی
  + یادگیری از خطاها و اصلاح پارامترها (گرادیان کاهشی)
* در واقع در یادگیری عمیق ابتدا وزن هر متغیر بصورت رندوم انتخاب می شود و سپس بر اساس آنها مقادیر محاسبه و نیز خطا محاسبه می گردد. سپس بر اساس میزان خطا و اختلاف آن با مقادیر واقعی، سعی می کنیم مینموم خطا را بدست آوریم که در اینجا از گرادیان کاهشی (مشتق تابع) استفاه می کنیم.
* روش کار به این صورت است که در یک نقطه اتفاقی (رندوم) مقدار تابع و خطا محاسبه می شود. اگر مقدار خطا زیاد باشد، نقطه انتخاب شده باید اصلاح شود تا به نقطه مینیموم تابع خطا برسیم. برای اصلاح مقدار **شیب نقطه ضربدر نرخ یادگیری (Learning Rate) را محاسبه** و از نقطه کم می کنیم. بدین صورت به نقطه مینیموم نزدیک می شویم.
* زمانیکه مقدار محاسبه شده بالا مساوی صفر و یا خیلی نزدیک به صفر باشد، گرادیان محو شونده (Vanishing) اتفاق می افتد.
* نکته مهم اینکه گرادیان کاهشی ممکن است دقیقا روی نقطه مینیموم قرار نگیرد. دلیل این امر این است که میزان حرکت از نقطه انتخابی اول با نرخ یادگیری مشخص می شود و ممکن است جایی در اطراف و نزدیک به نقطه مینیموم قرار گیرد. لذا انتخاب نرخ یادگیری در مدل اهمیت ویژه ای دارد.
* در انتخاب مقدار مینیموم و بدست آوردن بهترین نقطه، ممکن است مینیموم های محلی(Local) باعث ایجاد خطا در بدست آوردن مینیموم کلی (Global) شوند. به دلیل زیر در یادگیری عمیق این مورد اهمیت زیادی ندارد:
  + در شبکه های یادگیری عمیق معمولا تعداد متغیرها بسیار زیاد است و لذا معادلات دارای ابعاد (بعدها) زیادی هستند. این موضوع باعث می شود که نقطه مینیموم متغیرها در یک نقطه محلی اتفاق نیافتد و جایی که یک متغیر در مینیموم قرار گرفته است سایر متغیرها لزوما در مینیموم نباشند. مثلا نموداری که مشابه زین اسب باشد را در نظر بگیریم. در یکی از بعدها یعنی یک متغیر در مینیموم است ولی در متغیرهای دیگر در ماکزیموم قرار گرفته است.
  + بطور خلاصه می توان گفت که در شبکه های یادگیری عمیق اتفاق افتادن مینیموم محلی تقریبا ناممکن است.
* زمانی که عملکرد مدل خوب است، می توان با انجام کارهای زیر از درست بودن عملکرد و اطمینان از انتخاب نقطه مینیموم:
  + چندین بار مدل را آموزش دهیم و نتیجه را بررسی نماییم تا مطمئن شویم که عملکرد مدل درست است. در واقع در هر بار شروع چون از نقطه جدیدی بصورت اتفاقی کار آغاز می گردد، می تواند نقطه مینیموم را با دقت متفاوتی تخمین بزند.
  + ابعاد مساله را زیاد کنیم تا مطمئن شویم که نقطه مینیموم محلی وجود ندارد.
* **گرادیان** مجموعه ای از دو یا چند مشتق تابع از متغیرهای مختلف است. اگر متغیرهای معادله x, y, z باشند و از معادله نسبت به تک تک آنها مشتق گرفته شود و در یک لیست قرار گیرد، به این لیست گرادیان گفته می شود.
* در محاسبه گرادیان کاهشی، با تغییر نرخ یادگیری و نیز تعداد مراحل اجرای محاسبه مشتق و محاسبه نقطه جدید (epoch) می توان به جواب بهینه و نقطه مینیموم کلی رسید. البته اگر مدل دچار نقطه مینیموم محلی نشود (مدل هایی با تعداد متغیرهای کم).
* نرخ یادگیری می تواند بر اساس زمان (epoch) ، بر اساس شیب خط (مشتق تابع) متغیر باشد و یا بصورت ثابت در نظر گرفته شود.

گرادیان انفجاری (Exploding Gradient):

زمانی که شیب تایع (مشتق) خیلی زیاد شود، زمانیکه حالت عمودی به محور ایکس داشته باشد، مقدار محاسبه شده میزان تغییر شیب ضربدر نرخ تغییرات بسیار بزرگ می شود و وقتی این عدد را از نقطه ای که در آن قرار داریم کم یا زیاد کنیم، ممکن است از روی نقطه بهینه مینیموم پرش کنیم و آن نقطه را نبینیم. در این حالت گرادیان انفجاری رخ داده است.

فولدر 7 : Artificial Neural Network (ANN)

* مفهوم پرسپترون (Perceptron): متغیرهای ورودی شامل مقدار ثابت (Bias) و نیز مغیرهای مستقل بهمراه وزن مربوطه که مانند یک ضریب برای آنها عمل می کند وارد یک تابع می شوند که آنها را بصورت خطی با هم جمع می کند. سپس وارد یک تابع غیر خطی می شوند که تابع فعال ساز (Activation Function ) می باشد. و در نهایت یک پیش بینی متغیر هدف (وای هت) بعنوان خروجی خواهیم داشت. به مجموعه تابع خطی و تابع غیر خطی فعال ساز پرسپترون گفته می شود.
* مفهوم تابع فعال ساز (Activation Function): تابعی غیر خطی است که بر اساس ورودی هایی که به آن داده می شود، خروجی متفاوتی را بر می گرداند. معروف ترین و پرکاربردترین توابع فعال ساز، Relu ، Sigmoid و Hyperbolic tangent می باشند.
  + تابع Relu : برای مقادیر کمتر از صفر خروجی صفر دارد و برای مقادیر بیشتر از صفر همان مقدار را بر می گرداند. از این تابع معمولا در لایه های میانی استفاده می شود.
  + تابع Sigmoid : برای مقادیر منفی تا صفر خروجی 0 تا 0.5 می دهد و برای مقادیر بیشتر از صفر از 0.5 تا یک خروجی می دهد. از این تابع معمولا در لایه آخر (خروجی) استفاه می شود. در واقع برای دسته بندی باینری بسیار مناسب است و نتایج بیشتر از صفر در دسته یک و نتایج کمتر از صفر در دسته صفر قرار می گیرند.
  + تابع Hyperbolic tangent : برای مقادیر منفی تا صفر مقدار -1 تا صفر بر می گرداند و برای مقادیر مثبت از صفر تا مثبت یک بر می گرداند.
* مفهوم تابع خطا (Loss Function) : MSE جهت پیش بینی متغیرهای پیوسته مثل قیمت مسکن، دمای هوا و قد نفرات است استفاده می شود و Cross-Entropy (logistic) جهت متغیرهای گسسته مثل رد یا قبول شدن در امتحان، وجود یا عدم وجود تصویر حیوان در عکس، تحلیل احساسات منفی یا مثبت استفاده می شود.
  + MSE با فرمول زیر برای مقایسه متغیر هدف اصلی و متغیر هدف پیش بینی شده بکار می رود.
  + در محاسبه کراس آنتروپی احتمال رخ دادن یک پدیده (پیش بینی متغیر هدف) را با رخ دادن پدیده (متغیر هدف واقعی) مقایسه می کنیم. مقدار منفی در فرمول به دلیل لگاریتم مقادیر کمتر از یک می باشد که باعث ایجاد نتیجه منفی در داخل پرانتز می کند.
* مفهوم تابع هزینه (Cost function): وقتی تابع خطای متغیرها را با هم جمع و تقسیم بر تعداد کنیم یعنی میانگین توابع خطا را محاسبه کنیم به تابع هزینه دست پیدا می کنیم. در واقع تمام متغیرهای هدف پیش بینی شده را با یکی از روش های فوق با مقدار متغیر هدف واقعی مقایسه و تابع خطا را بدست می آوریم و میانگین آنها را بدست می آوریم.
* هدف از محاسبه توابع هزینه و خطا محاسبه وزن های هر متغیر و در نتیجه کاهش حداکثری این توابع (هزینه و خطا) می باشد.
* استفاده از تابع خطا برای بهینه سازی ممکن است باعث ایجاد حجم زیاد محاسبات و نیز ایجاد اورفیت در مدل شود. لذا از تابع هزینه جهت انجام محاسبات استفاده می نماییم و این کار را بصورت بچ به بچ انجام می دهیم.
* به مجموعه ضرایب مربوط به ورودی ها و نیز جمع خطی آنها (سیگما) و اعمال تابع فعال سازی (اکتیویشن) یک نود (node) گفته می شود.
* هر نود ممکن است چندین **ورودی** داشته باشد ولی **فقط یک خروجی** دارد که برای ورود به نودهای دیگر در لایه های بعدی به تعداد مورد نیاز کپی می شود.
* هر نود بصورت کاملا مجزا از بقیه کار می کند و کاملا منحصر بفرد است. در واقع کارکرد نود ها روی همدیگر اثری ندارد و فقط ورودی ها هستند که بر خروجی نود اثر گذار هستند.

**مفهوم حرکت برگشت (Back Propagation)**

* زمانیکه ضرایب متغیرها بصورت اتفاقی انتخاب و بر اساس آن پیش بینی ارائه می گردد، نیاز است تا به سمت بهترین و کمترین خطای ممکن حرکت کنیم. این تغییر بصورت گرادیان کاهشی اتفاق می افتد که از نظر مفهومی با حرکت برگشتی معادل است. در واقع حرکت ما به عقب بمنظور اصلاح **ضرایب متغیرها** بوسیله روش گرادیان کاهشی را حرکت برگشت گویند.
* یادآوری اینکه در گرادیان کاهشی مقدار متغیر به اندازه نرخ تغییر (شیب یا مشتق) ضربدر نرخ یادگیری کاهش یا افزایش می یافت. در اینجا هم **میزان تغییر ضرایب** به اندازه شیب یا مشتق تابع نسبت به ضریب ضربدر نرخ یادگیری تغییر (افزایش یا کاهش) می یابد.
* مقدار مشتق یا شیب تابع نسبت به ضرایب به **نوع تابع خطا** (یا تابع هزینه) و **تابع فعال سازی** بستگی دارد.

حل مساله رگرسیون خطی بوسیله یادگیری عمیق

* در این مسایل چون از تابع فعال سازی غیر خطی استفاده می شود، ممکن است نتایج بصورت یک خط کامل بدست نیاید و در بعضی نقاط غیر خطی دیده شود.

حل مساله دسته بندی باینری با ANN

* در حل این مسائل باید دقت شود که لزوما بهترین نتیجه از روش های یادگیری عمیق بدست نمی آید و شاید استفاده از روش های یادگیری ماشین مثل KNN یا Kmeans نتیجه بهتری داشته باشند.
* تغییر در نرخ یادگیری می تواند نتایج بسیار متفاوتی را در مدل ایجاد کند. البته با توجه به ماهیت اختصاص اتفاقی ضرایب به متغیرها، ممکن است تغییرات نرخ یادگیری در یک بار آموزش و اجرای کد نتیجه ندهد و لازم باشد که بارها تکرار شود تا بهترین نتیجه بدست آید.
* با تغییر نرخ تغییر در مدل، دقت (accuracy) از 50 درصد تا حدود 99 درصد تغییر می کند.
* استفاده از توابع غیر خطی فعال ساز در مسایل خطی و بطور کلی استفاده از روش های پیچیده در حل مسائل ساده باعث عدم بازدهی مدل های می گردد.
* بطور خلاصه برای مسائل ساده باید از روش های ساده استفاده کرد.
* با حذف توابع غیر خطی فعال ساز، دقت مدل از 99.5 تا 100 درصد ارتقا می یابد. این در حالیست که کلیه پارامتر های مدل و نرخ یادگیری و ... همانند سابق باقی می مانند.

استفاده از لایه های داخلی (پنهان)

* در پردازش داده ها در شبکه های عصبی می توانیم لایه ها را بصورت جفت خطی و غیر خطی (تابع فعال سازی) در نظر بگیریم و به مدل اضافه نماییم.
* تعداد ورودی مدل به تعداد خصوصیات (فیچرها) بستگی دارد و تعداد لایه های داخلی به دلخواه انتخاب می شود. خروجی لایه اول بعنوان ورودی لایه بعدی می باشد و باید تعداد ورودی لایه بعدی با خروجی لایه قبلی یکسان باشد. خروجی لایه آخر با توجه به نوع مساله مشخص می شود.
* معمولا در لایه های میانی از تابع فعال ساز Relu استفاده می شود و در لایه آخر و خروجی نهایی مدل از Sigmoid استفاده می کنیم.
* اگر در مسائل دسته بندی از تابع سیگموید استفاده گردد، تابع خطا BCELoss در نظر گرفته می شود.
* می توانیم بجای استفاده از BCELoss از تابع BCELogitsLoss استفاده کنیم و در این حالت باید تابع سیگموید را از مدل حذف نماییم.

نکته مهم:

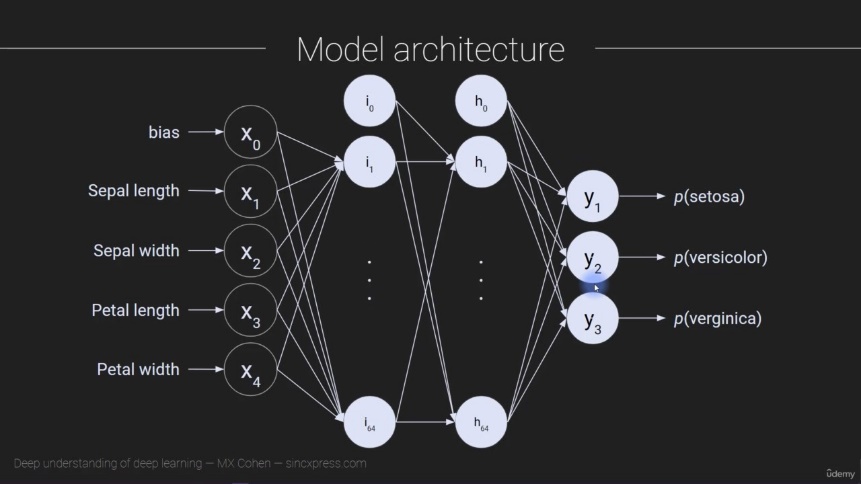
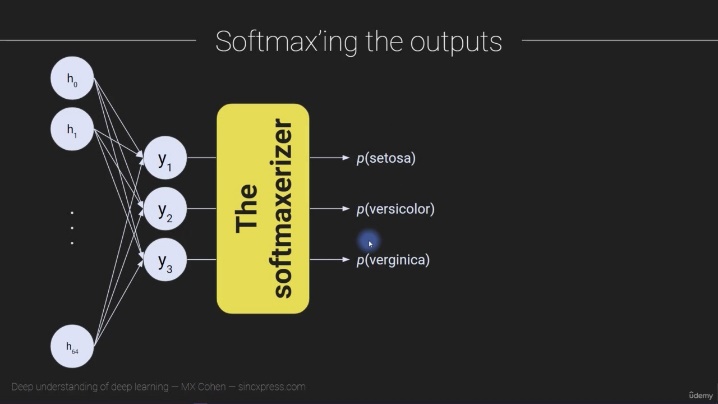
در حل مسائل غیر خطی (Non-Linear) پیچیده از مدل های یادگیری عمیق استفاده می نماییم. اینکه می توانیم و دانش ایجاد مدل های یادگیری عمیق را داریم به این معنی نیست که همه مسائل را در این چارچوب و مدل ها حل کنیم.

Just because you can, doesn’t mean you should.

برای حل هر مساله بهتر است بهترین و خلاقانه ترین و در عین حال ساده ترین مدل ها را در نظر بگیریم.

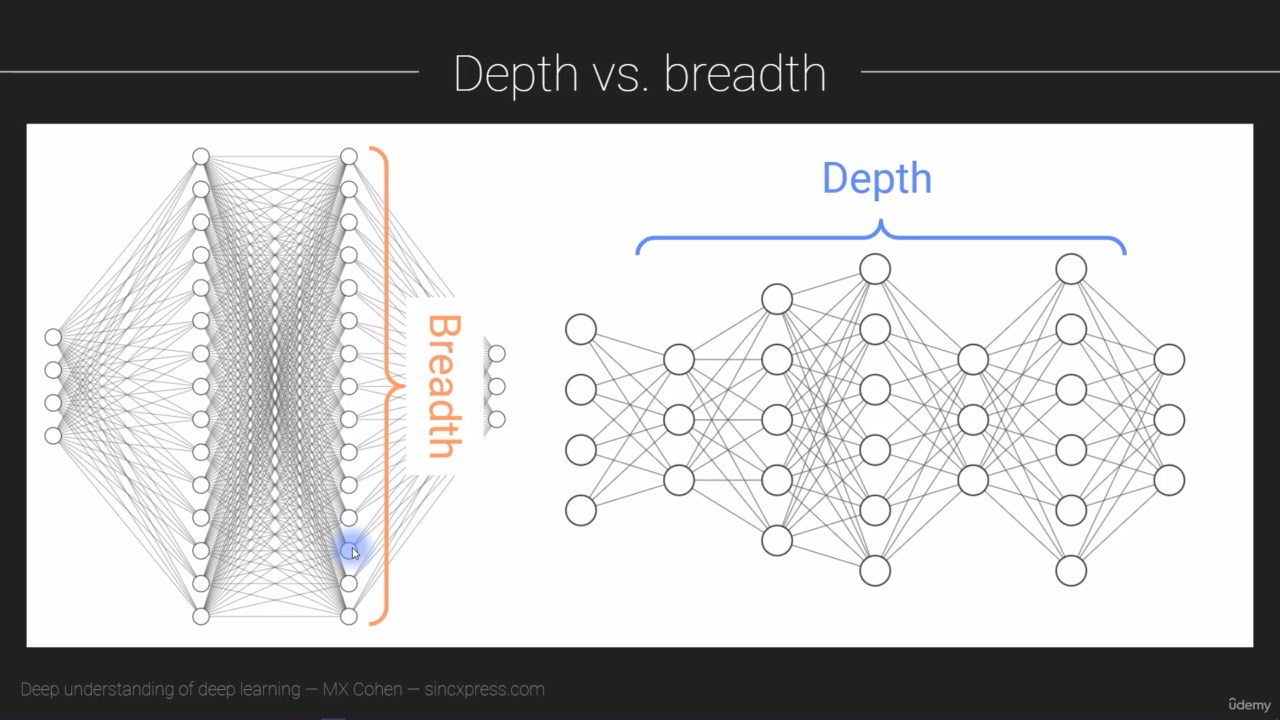
* مدل های حل مسائل خطی فقط یک لایه دارند. در واقع به دلیل ماهیت الگوهای معادلات خطی، لایه های میانی در عملیات ریاضی حذف خود به خود می شوند.

حل مساله دسته بندی گل های Iris

* در این مساله از روش یادگیری عمیق استفاده می کنیم. چهار خصوصیت (فیچر) داریم و دسته بندی نهایی در سه دسته خواهد بود.
* نکته در مورد bias: مقدار بایاس در تمام لایه ها برابر یک می باشد و ورودی از لایه قبلی ندارد و تنها به لایه بعدی وارد می شود.
* در خروجی این مدل که سه دسته از گل ها باید مشخص شوند، نمی توانیم از تابع فعال سازی سیگموید استفاده کنیم چراکه در این تابع فقط دو خروج بصورت صفر و یک خواهیم داشت.
* در این مدل از سافت مکس (Softmax) استفاده می شود که خروجی این تابع مقدار احتمال رخ داد می باشد.
  + فرمول محاسبه احتمال در سافت مکس برای هر i برابر است با :
* 
* در واقع سافت مکس بیشترین احتمال را به بیشترین مقدار ورودی می دهد.

پهنا و عمق مدل یادگیری عمیق (Depth and Breadth or width)

* تعدا نود ها در یک لایه را پهنا (Breadth or width) و تعداد لایه ها را عمق (Depth) مدل گویند.

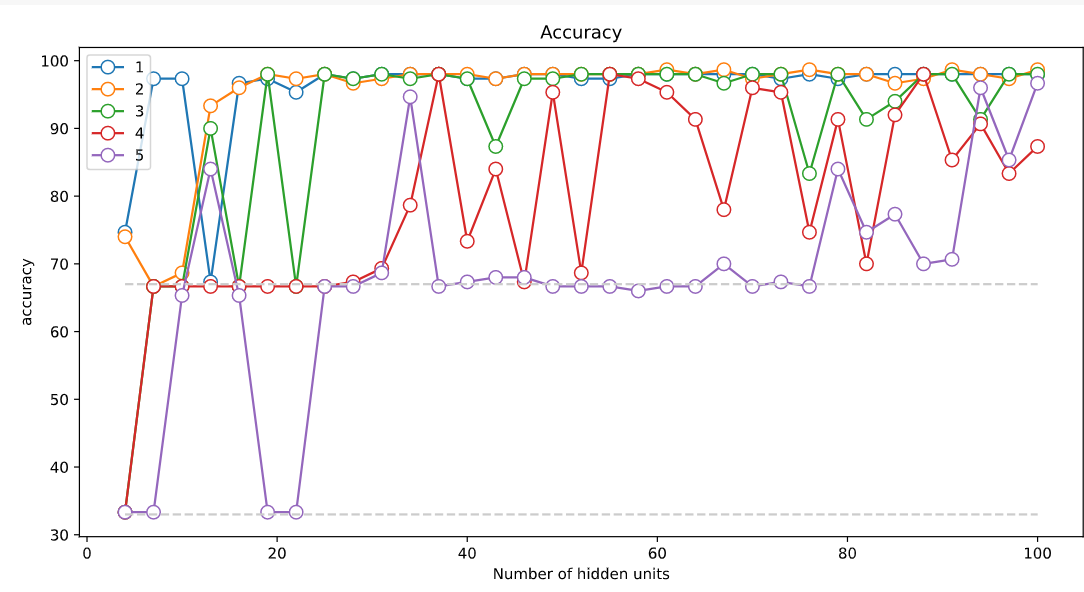


* تعدا پارامترهای یادگیری (تعداد فلش ها که نشان دهنده وزن متغیرها هستند) در مدل هایی با تعداد نود های یکسان که در پهنا و عمق با هم متفاوت هستند تفاوت می کند.

See the following code file

ANN\_01.ipynb

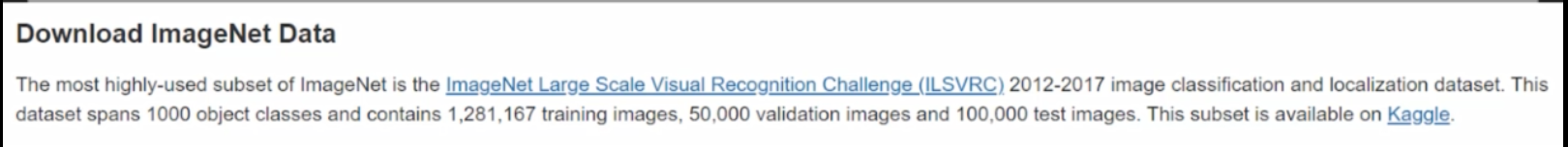
* نکته اینکه تعداد لایه های عمق یا پهنا لزوما باعث ایجاد شرایط دقیق تر نخواهند شد. در واقع افزایش تعداد نودها در لایه یا افزایش تعداد لایه ها لزوما باعث بهتر شدن کارآیی مدل نمی شوند.
* نمودار زیر نشان می دهد که افزایش تعداد نود ها در لایه تاثیر چندانی در سرعت یادگیری یا عملکرد مدل و دقت آن ندارد. ضمن اینکه تعداد لایه ها که با رنگ های مختلف در نمودار مشخص شده است نشان می دهند که مدل با تعداد لایه های کمتر می تواند سریعتر به دقت بالاتر برسد.



* بطور خلاصه می توان گفت مدل هایی با عمق کم سرعت یادگیری بیشتری دارند و مدل هایی با عمق بیشتر می توانند موارد و مسائل پیچیده تر را حل کنند.
* کارآیی مدل لزوما به تعداد پارامترهای یادگیرنده آن (وزن ها یا فلش ها) ربطی ندارد.

فولدر 8 : اورفیت و کراس ولیدیشن (Overfitting and Cross-Validation)

* تعداد پارامترهای مدل را چگونه تعیین کنیم؟
  + اگر مساله دارای یک یا دو پارامتر باشد، می توان با مشاهده داده ها در نمودار روابط ریاضی بین آنها را بطور چشمی تقریب زد.
  + اگر مساله دارای ابعاد و پارامتر های بیشتری باشد، روش چشمی و ترسیم نمودار مفید و قابل استفاده نخواهد بود. در این مسائل از روش های آماری برای تعیین تعداد پارامتر ها استفاده می کنیم. یکی از این روش ها **کراس ولیدیشن** است
* اورفیت امکان تعمیم دادن مدل بر روی داده های جدید را ضعیف می کند ولی اگر بخواهیم از همان داده ها و همان شرایط استفاده کنیم، مدل بهتری خواهد بود. در واقع اگر نخواهیم که داده های جدید به مساله بدهیم و پیش بینی بگیریم بحث اورفیت مشکلی ایجاد نمی کند.
* تعمیم (Generalization) و حدود آن (Generalization Boundaries) در بحث اورفیت بسیار مهم هستند. بطور مثال اگر بخواهیم قیمت مسکن در شهری را ارزیابی و مدل آن را تهیه نماییم شاید نتوانیم با همان مدل قیمت مسکن در شهر دیگری را ارزیابی یا پیش بینی نماییم. لذا حدود مدل ما محدود به شهر اول می باشد.
* دیتاست مربوط به imagenet



* استفاده از تابع train-test-split در کتابخانه سایکیت لرن و سپس بکارگیر مدل ANN روی آن جهت داده های گل های Iris

فولدر 9 : Regularization

فولدر 10 : 10 Metaparameters (activations, optimizers)

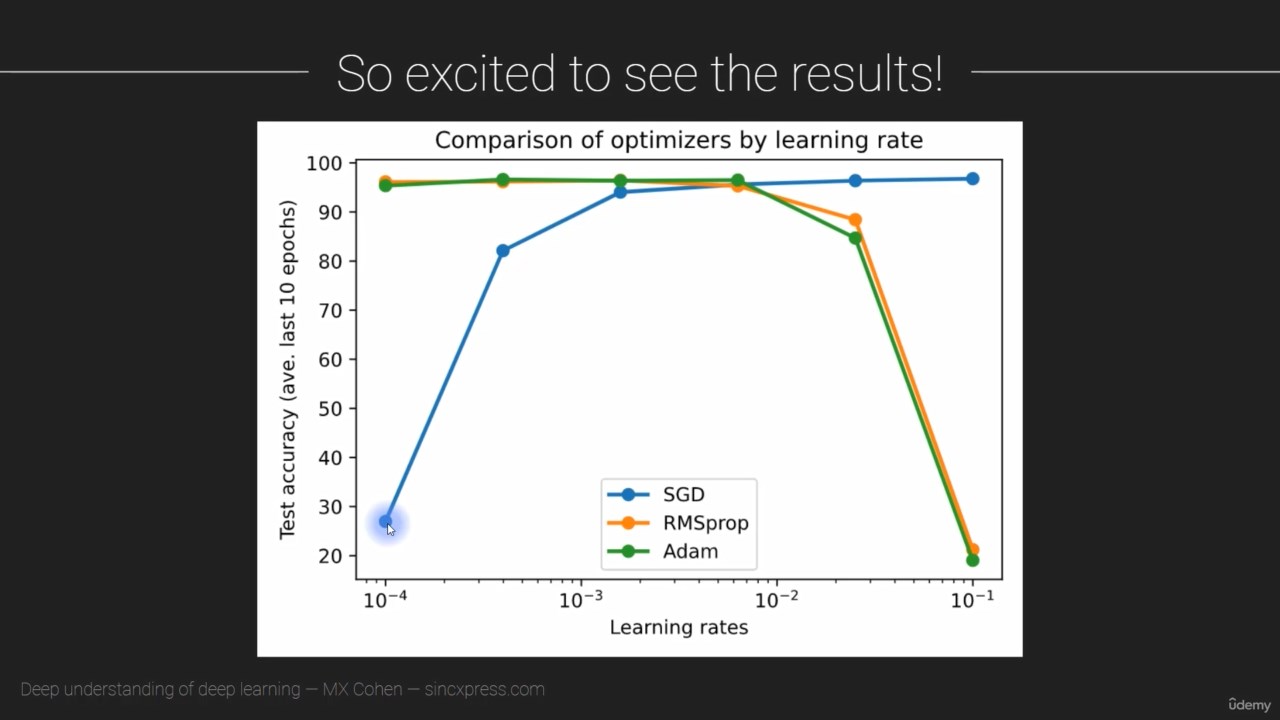
فولدر 11 : FFN

دیتاست MNIST

file : MNIST\_1400\_12\_04.ipynb

* مقایسه انواع بهینه ساز ها (Optimizers) با اختصاص نرخ یادگیری های متفاوت

همانطور که در نمودار دیده می شود، استفاده از نرخ یادگیری کمتر در گرادیان کاهشی اثر منفی دارد ولی در دو بهینه ساز دیگر اثر خوبی گذاشته است. دلیل این موضوع این است که در بهینه ساز آدام و آر ام اس پراپ روش دینامیک برای تغییر نرخ یادگیری پیش بینی و تعبیه شده است.



فولدر 12 : More on Data

در این بخش استفاده از انواع دیتاست ها و نیز استفاده از داده های دیتاست های آنلاین و وارد کردن اطلاعات به گوگل کولب و طریقه استفاده از آنها بررسی می شود.

* در فرآیند استفاده از داده ها در پایتورچ، ابتدا باید داده ها را از قالب نامپای به تنسور تبدیل کرد. سپس باید داده ها و لیبل ها را بوسیله دستور TensorDataset به دیتاست تبدیل کرد.
* استفاده از دیتا در ورودی مدل های شبکه های عصبی اگر بصورت بچ به بچ صورت پذیرد باعث افزایش چشمگیر راندمان و کارکرد این شبکه ها می گردد لذا از دستور dataloader برای این کار یعنی وارد کردن داده ها بصورت بچ به بچ و قسمت بندی شده استفاده می کنیم.
* دیتالودر داده ها را دسته بندی می کند ولی بصورت یک شی قابل استفاده در چرخش ها (Iterable) قابل استفاده است و در آن صورت می تواند داده ها را به مدل وارد نماید.
* داده هایی که تعداد آنها کم است و برای آموزش مدل کفایت نمی کند، و یا نوع داده ها همخوانی ندارند مثل جداسازی تصاویر گربه از قایق نیاز به انجام کار اضافی روی داده ها داریم. در این شرایط می توان از روش های زیر استفاده نمود:
  + داده های بیشتری بدست آوریم..... گاهی اوقات شدنی نیست.
  + تعدادی از داده های بیربط را حذف کنیم.....زمانیکه داده های زیادی داریم که داده های پرت در آنها وجود دارد
  + از داده های موجود بارها و بارها بصورت تکراری نمونه برداری نماییم..... احتمال اورفیت شدن را زیاد می کند.
  + تقویت داده ها (Data Augmentation) به این معنی که خصوصیات غیر خطی به داده ها اضافه نماییم....مثل لگاریتم گرفتن یا پردازش هایی روی تصاویر بعنوان داده های ورودی
  + تولید داده های مشابه داده های فعلی....

SMOTE: Synthetic Minority Oversampling Technique

* + گاهی اوقات ممکن است بتوانیم از روشی غیر از شبکه های عصبی و یادگیری عمیق، مساله را حل کنیم. داده های نامتوازن در شبکه های عصبی می تواند مخرب باشد و لذا شاید بهتر باشد از انواع دیگر مدل های ماشین لرنینگ استفاده کنیم.
* در تقویت داده ها (Data Augmentation) باید دقت کنیم که لزوما با اضافه کردن یک خصوصیت ترکیبی از داده ها، خصوصا اگر این خصوصیت بصورت خطی اضافه شود، ممکن است در کارکرد مدل تاثیر گذار نباشد. اگر خصوصیت ترکیبی غیر خطی اضافه شود ممکن است شرایط بهتر باشد ولی لزوما به معنی رفع مشکلات قبلی نیست.

فولدر 15: Weight inits and investigations

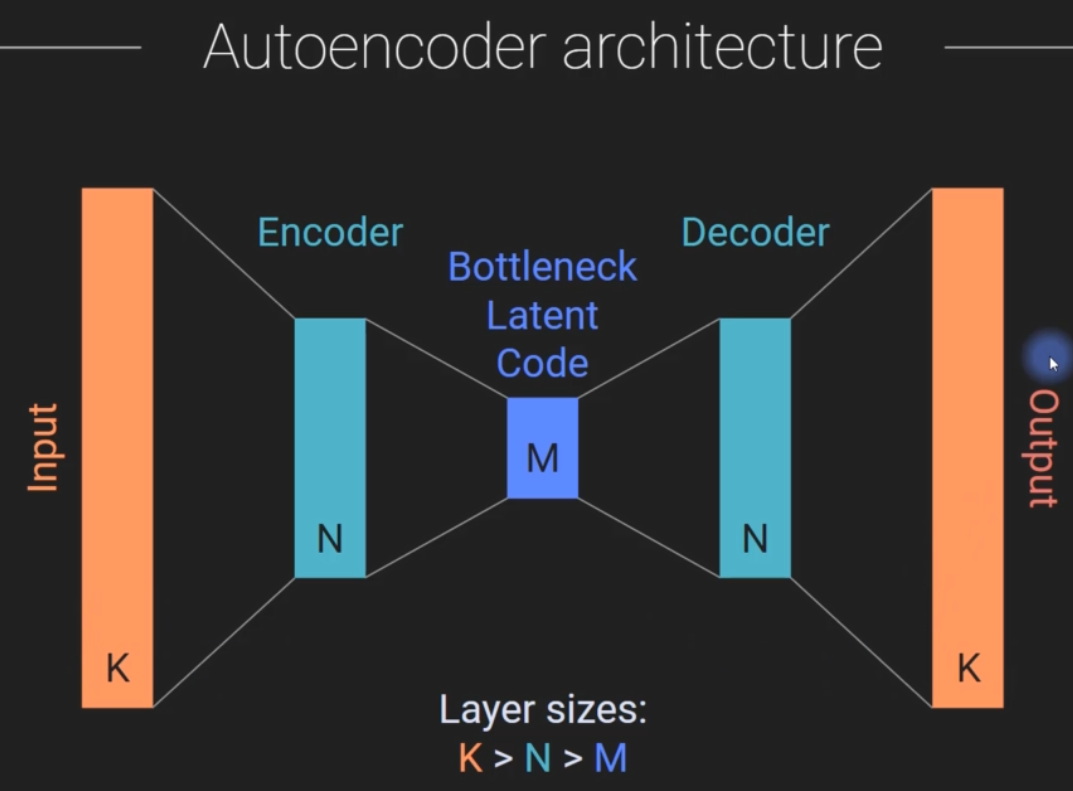
* ماتریس های وزن متغیرها در مدل های شبکه های عصبی بر مبنای تعداد ورودی و خروجی ها مشخص می شوند. با توجه به نوع معادلات جبری که در این شبکه ها نوشته می شود و نیاز به انجام ضرب ماتریسی متغیر ها و ضرایب، در صورت فراخوان کردن ماتریس ضرایب در مدل (پای تورچ) ابتدا تعداد خروجی (پارامتر اول لیست یا تنسور) و سپس تعداد ورودی (پارامتر دوم لیست یا تنسور) نمایش داده می شود. مثلا اگر مدل دارای 3 ورودی و لایه میانی با 5 و 8 نود و خروجی 4 تایی باشد، در صورت فراخوان ماتریس ضرایب (weights) خروجی بصورت زیر نمایش داده می شود:

(5, 3) , (8, 5), (4, 8)

* اختصاص وزن اولیه به مدل :
  + در صورتیکه وزن های اولیه ای به مدل اختصاص دهیم، باید دقت کرد که اعداد مختلف بصورت رندوم باشد. در حالتیکه از اعداد یکسان استفاده شود، هیچ مدل یادگیری عمیقی نمی تواند درست عمل نماید.
  + در مسائل نسبتا ساده اختصاص وزن اولیه در مدل یادگیری عمیق کمک زیادی به کارکرد مدل نمی کند ولی در مسائل بزرگ با داده های بسیار زیاد بهتر است این کار انجام شود.

فولدر 16 : Autoencoders

* منظور از اتوانکودر سیستمی است که بتواند خود را برای انکود کردن (معمولا ساده سازی یا کاهش بُعد) داده ها آموزش دهد.
* ساختار اتوانکودر به صورت زیر است:



فولدر 17 : Running models on GPU

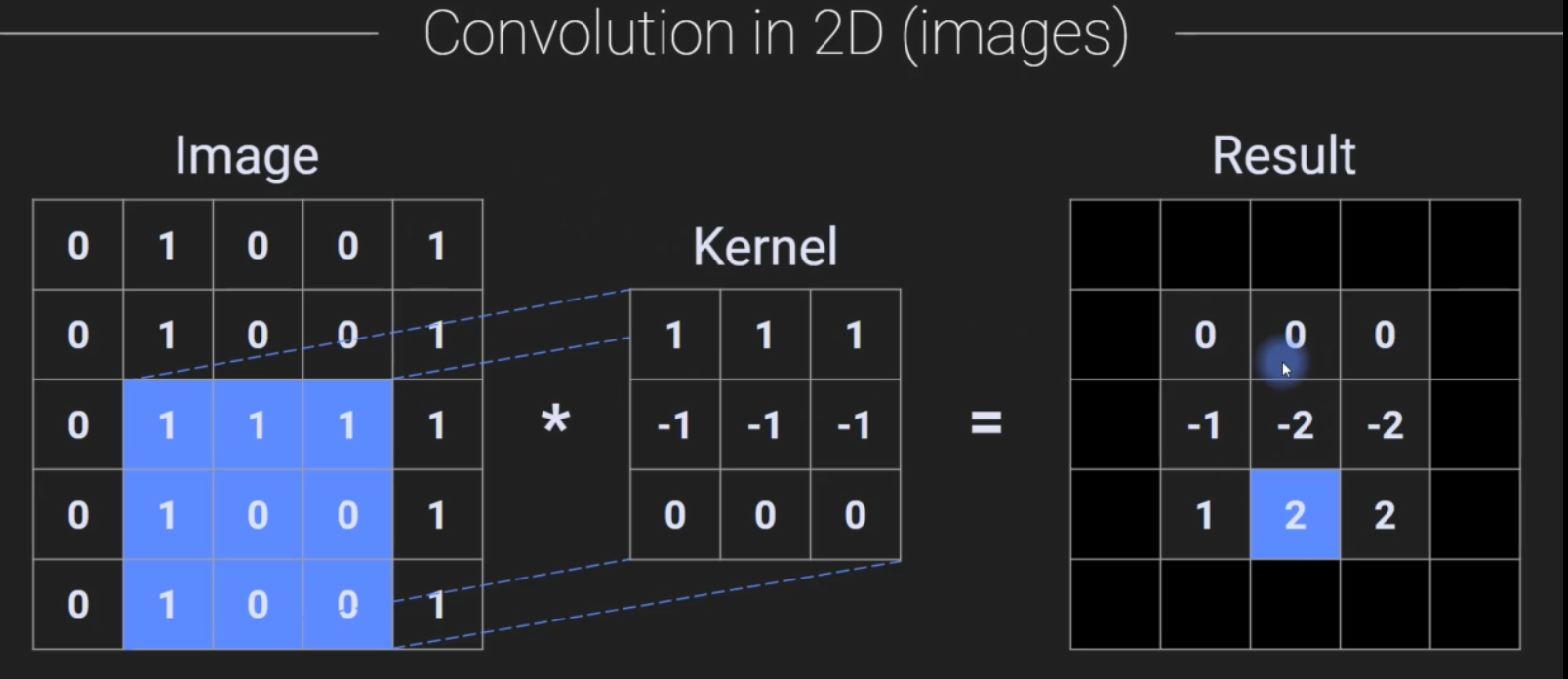
* برای استفاده از جی پی یو، علاوه بر استفاده از سخت افزار باید در کدهای برنامه و نرم افزار نیز استفاده از آن قید گردد.
* در پای تورچ از دستورات زیر برای بکار گیری جی پی یو استفاده می شود.
* # use GPU
* device = torch.device('cuda:0' if torch.cuda.is\_available() else 'cpu')

در واقع زمانی که cuda که نشانگر امکان استفاده از جی پی یو است درست باشد از جی پی یو استفاده می کند و در غیر اینصورت از سی پی یو استفاده می شود.

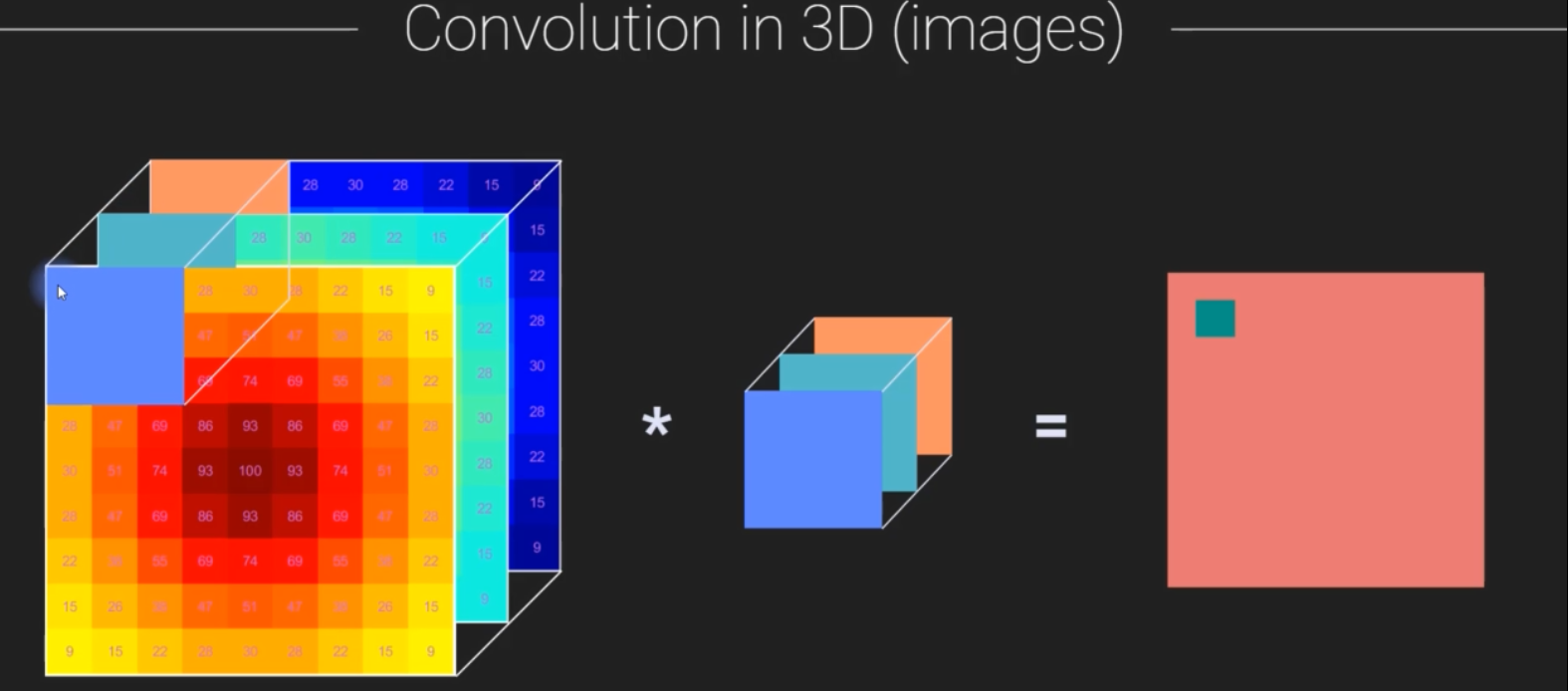
* در برنامه های کوچک و کم حجم استفاده از جی پی یو ارجحیت چندانی ایجاد نمی کند.
* بهتر است از جی پی یو سایت هایی مثل گوگل کولب و یا AWS یا AZURE استفاده نماییم که در حال حاضر یکی از بهترین ها گوگل کولب است.
* در صورتیکه زیاد و نابجا از جی پی یو گوگل کولب استفاده نماییم ممکن است برای ساعاتی یا چند روزی دسترسی را محدود سازد ولی دوباره برقرار می شود.
* در حین انجام فرآیند یادگیری مدل، هرجا که لازم باشد می توانیم با وارد کردن آرگومان cpu در پارامتر device در دستورات پای تورچ بجای جی پی یو از سی پی یو استفاده نماییم.

فولدر 18 : Convolution and transformations

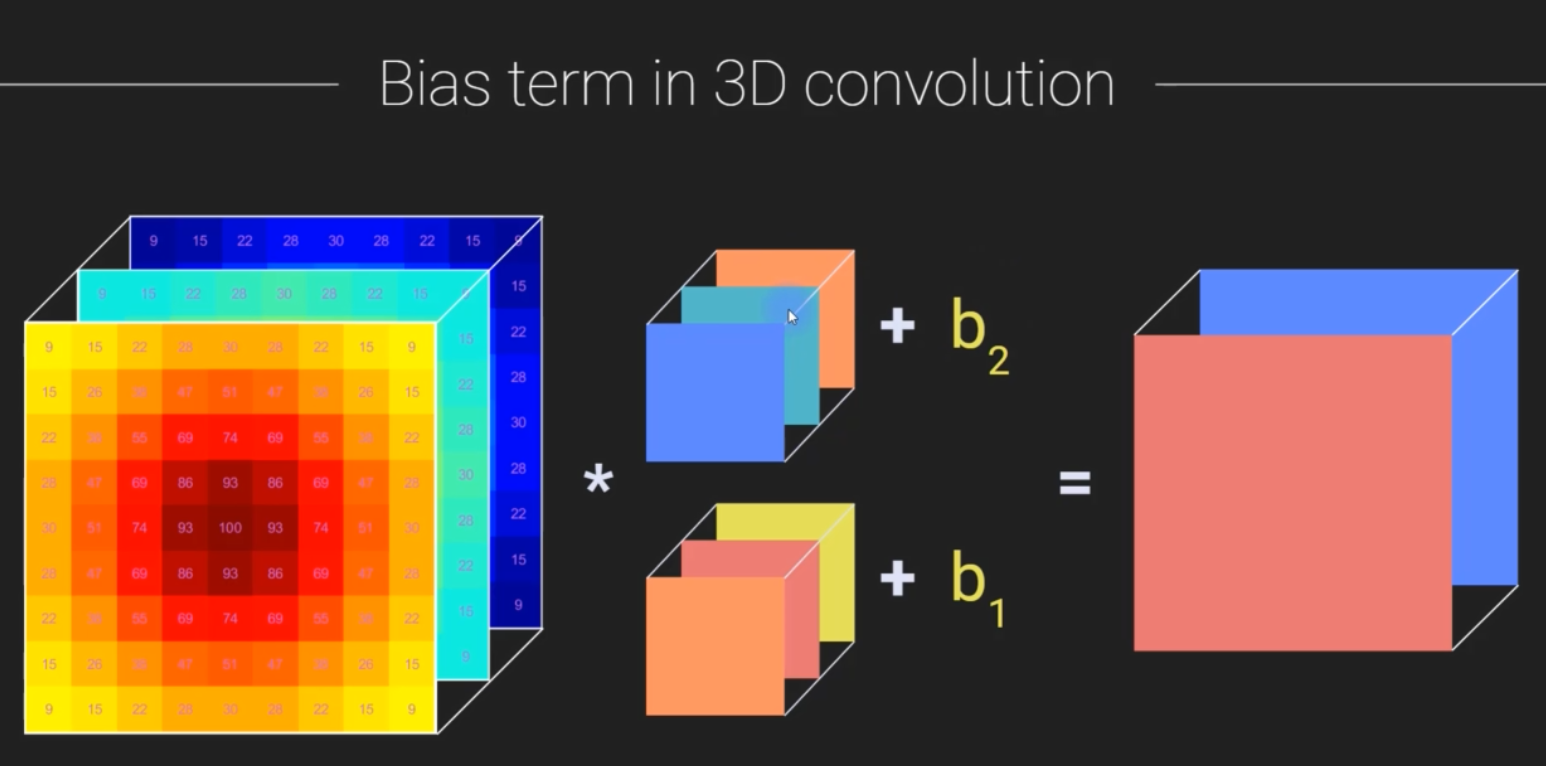
* منظور از کانولوشن استفاده از یک ماتریس ضرایب به نام کرنل (kernel) و ضرب ماتریسی (دات پروداکت) آن در ماتریس داده ها می باشد.
* برای آنکه عملیات ضرب ماتریسی بدرستی صورت پذیرد می توان کناره های ماتریس بزرگتر را با اعداد صفر اضافه کرد (padding or zero padding) تا ماتریس حاصلضرب نهایی با ابعاد مشابه ماتریس اصلی بدست آید.
* در تصویر زیر پدینگ صورت نگرفته است و لذا ماتریس حاصلضرب به اندازه ماتریس کرنل است.



* ماتریس های تصاویر رنگی که حاوی لایه های رنگ قرمز و سبز و آبی هستند با سه ماتریس متناظر کرنل ضرب و به آن کانولوشن سه بعدی گویند.



* می توان از تعداد بیشتری کرنل های تبدیل استفاده کرد. بعنوان مثال در تصاویر می توان از دو کرنل برای تبدیل لایه های رنگی تصاویر استفاده کرد و هر کدام از آنها یک لایه جدید ایجاد می نمایند.



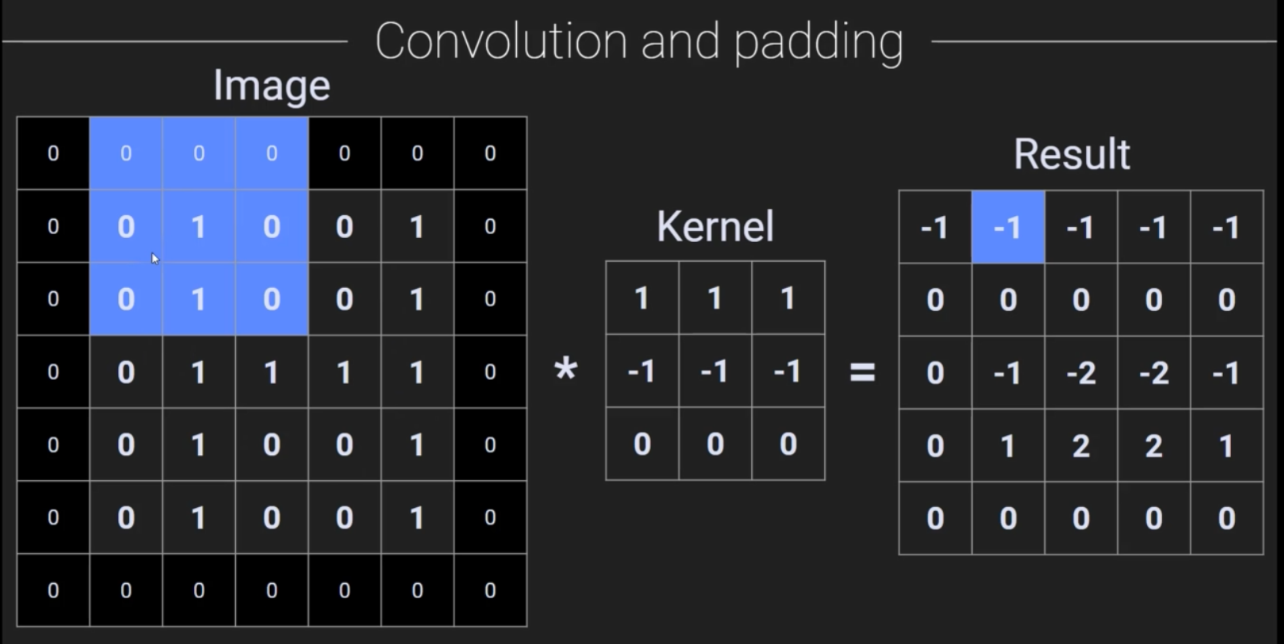
* در کنار ماتریس ضرایب کرنل یک عدد قابل یادگیری ثابت بایاس (Bias) نیز وجود دارد که می توان بعنوان یکی از پارامترهای سیستم از آن یاد کرد.
* حاصلضرب ماتریس های سه بعدی بالا در تعداد کرنل های مشخص، باعث ایجاد همان تعداد (به تعداد کرنل ها) ماتریس می شود که به آنها کانال (Channel) گفته می شود. در واقع اعداد این ماتریس ها خصوصیاتی هستند که محاسباتی روی آنها انجام شده است و با رنگ های قرمز و سبز و آبی (RGB) متفاوت هستند.
* در واقع سه کانال اصلی و اولیه تصاویر کانال های قرمز و سبز و آبی هستند که با ضرب در ماتریس ضرایب کرنل تبدیل به کانال های جدیدی می شوند.
* با توجه به اینکه عدد حاصلضرب ماتریس کرنل در ماتریس داده ها بصورت یک عدد بدست می آید، معمولا ابعاد ماتریس ضرایب کرنل را بصورت عدد فرد انتخاب می کنند (3 و 5 و 7و ..) که مرکز یکتایی داشته باشد و دقیقا در مرکز ماتریس واقع شود.
* به ماتریس های ضرایب کرنل ماتریس های فیلتر یا فیلتر کانولوشن نیز گفته می شود.
* در واقع برای تصاویر مختلف از کرنل های ثابت استفاده می شود و لذا پس از ضرب آنها در ماتریس داده های تصاویر خصوصیات جدید (Features) در تصاویر دیده می شود.
* در فرآیند یادگیری عمیق ماتریس ضرایب بصورت اتفاقی (رندوم) در نظر گرفته می شوند و سپس در طول فرآیند یادگیری ضرایب تغییر می نمایند.
* پس از انجام فرآیند یادگیری، کرنل ها برای همه تصاویر ثابت می مانند.
* استفاده از مدل هایی که کرنل ها در آنها ایجاد شده اند و اصطلاحا از قبل یادگیری شده اند (Pre Trained) به Transfer Learning معروف است.
* استفاده از کرنل ها در فرآیند یادگیری عمیق بمنظور استخراج خصوصیات جدید است و نه بمنظور دسته بندی یا تصمیم گیری برای داده های ورودی. در واقع از خصوصیات جدیدی که بدست می آید می توان برای یادگیری مدل استفاده نمود.
* ابعاد تنسور های مدل در پای تورچ مساوی حاصلضرب تعداد کانال ها در تعداد پیکسل های عرض در تعداد پیکسل های ارتفاع تصویر می باشد.
* مثلا اگر تصویر رنگی بعلاوه دو کانال جدید که خطوط عمودی و افقی را مشخص نموده است داشته باشیم، ابعاد تنسور ورودی مدل مساوی 5x1675x3000 = 25125000 خواهد بود.

**استفاده از سای پای (Scipy convolve2d) و پای تورچ (F.conv2d) در فیلتر کردن تصاویر**

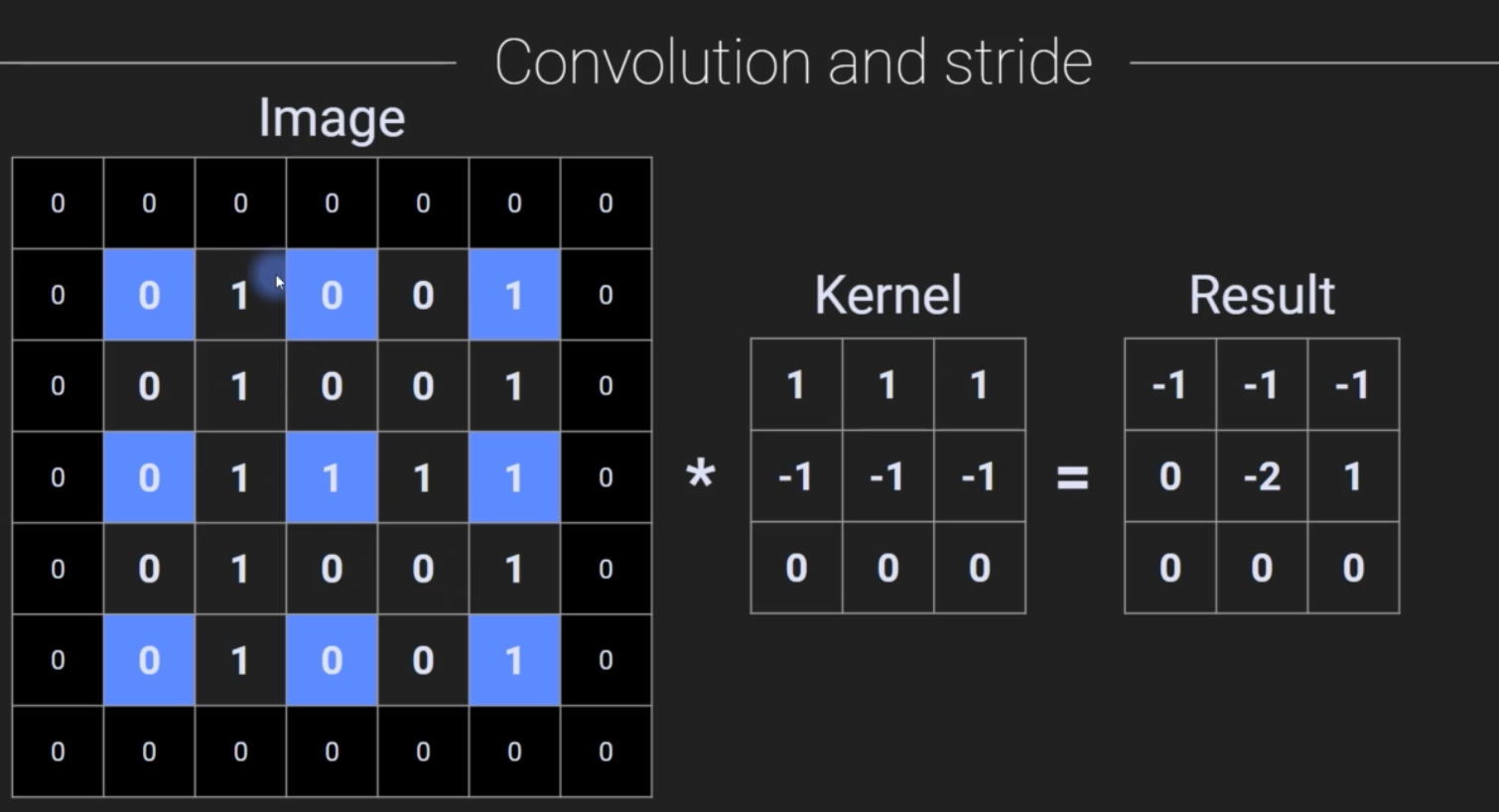
* **فایل Convolution\_1400\_12\_05.ipynb**
* با استفاده از توابع بالا می توان بصورت ساده ماتریس ضرایب کرنل را در تصاویر اعمال (ضرب) کرد.
* سای پای کناره های تصویر (پدینگ) را حذف می کند.

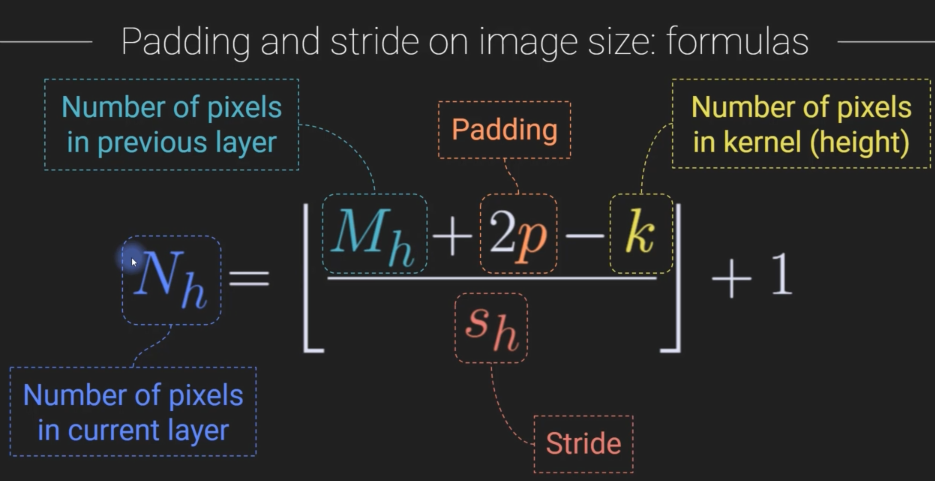
مفهوم پدینگ (Padding)

* استفاده از یک ردیف در هر چهار طرف تصویر باعث می شود تصویر خروجی یعنی حاصلضرب ماتریس کرنل در تصویر اصلی با ابعاد اولیه برابری نماید. تصویر زیر نشان می دهد که یک ردیف در هر وجه با مقدار صفر به ماتریس داده های تصویر اولیه اضافه شده است.



مفهوم استراید (Stride)

* اگر لازم باشد تا ابعاد تصویر کاهش یابد یکی از راهکارها کم کردن تعداد پیکسل های تصویر است. روش استراید کمک می کند تا پیکسل ها را یکی در میان یا چند تا در میان در ماتریس کرنل ضرب کنیم و در نهایت ماتریس حاصلضرب ابعاد کمتری نسبت به تصویر اصلی خواهد داشت.
* در تصویر بالا ابتدا پدینگ و سپس استراید انجام شده است و در نهایت ماتریس خروجی از یک تصویر پنج در پنج و کرنل سه در سه یک ماتریس سه در سه می باشد.
* کاهش ابعاد تصویر باعث کاهش پارامتر های یادگیرنده در شبکه های یادگیری عمیق CNN می شود.
* برای محاسبه اندازه تصویر در یکی از لایه های شبکه عصبی بر اساس میزان پدینگ و استراید از فرمول زیر استفاده می کنیم:



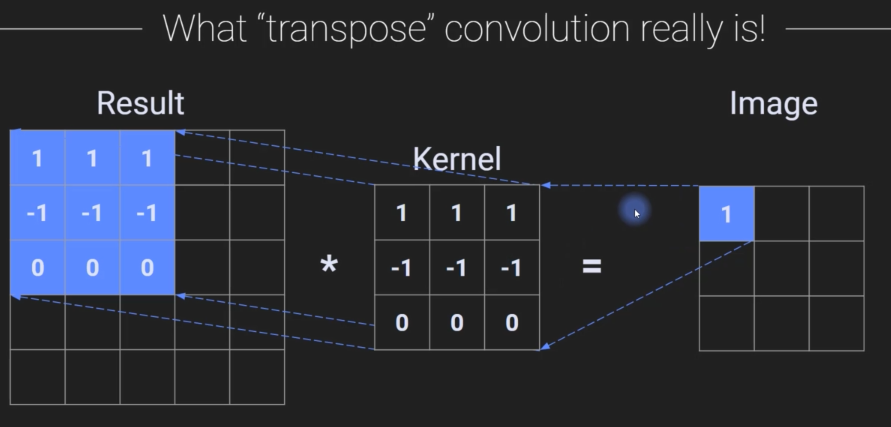
* در این فرمول h به تعداد پیکسل های ارتفاع (ردیف) اشاره دارد.
* وجود براکت های ال شکل (L) در اطراف فرمول به معنی روند کردن به سمت پایین می باشد. یعنی استفاده از جزء صحیح عدد بدست آمده + یک.

استفاده از تابع کانولوشن پای تورچ Conv2

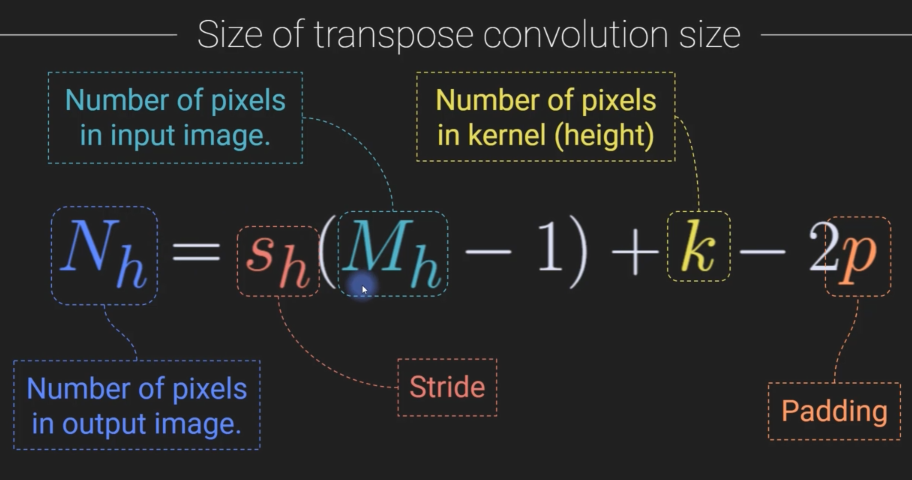
* **فایل Convolution\_1400\_12\_05.ipynb**

**مفهوم Transposed Convolution**

* این عبارت ممکن است این خطا را در ذهن بوجود آورد که ماتریس ضرایب را بصورت ترنسپوز درآورده و حاصلضرب را محاسبه می کنیم **ولی این درست نیست**.
* در واقع روش محاسبه کانولوشن ترنسپوز این است که برعکس روش پیشروی عمل می کنیم و یک عدد را با استفاده از کرنل به یک ماتریس تبدیل می کنیم. مشابه تصویر زیر.



* این روش بیشتر برای افزایش تعداد پیکسل ها و دقت و رزولوشن در اتوانکودر ها استفاده می شود.
* برای بدست آوردن ابعاد تصویر در لایه ها از فرمول زیر استفاده می نماییم:

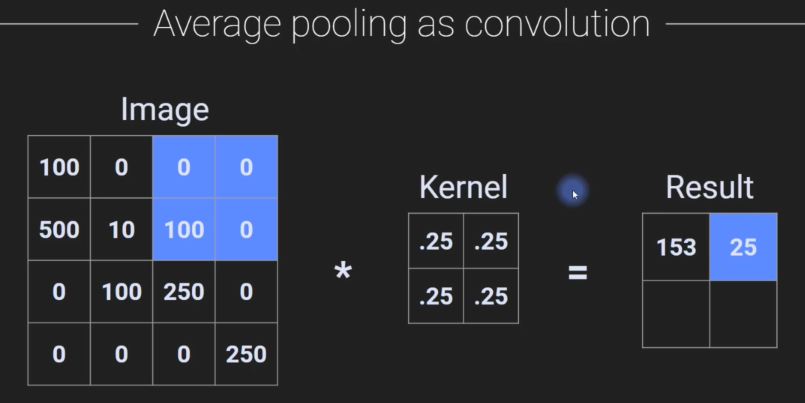


* در پای تورچ از تابع Conv2Transpose2d برای انجام کانولوشن ترنسپوز استفاده می شود

**مفهوم Max/mean Pooling**

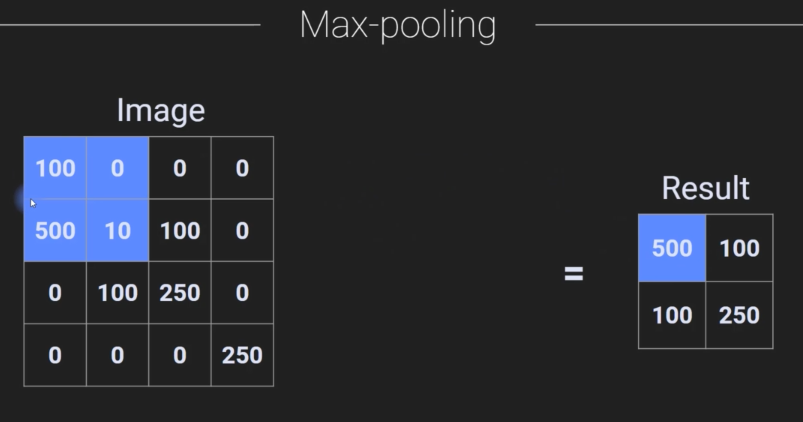
**Average Pooling**

* اگر از یک ماتریس کرنل که محتوای آن میانگین یک عدد باشند مثلا یک ماتریس دو در دو که همه اعضای آن 0.25 باشند (عدد یک تقسیم بر چهار) برای تبدیل کانولوشن استفاده نماییم، آنگاه در ماتریس حاصلضرب میانگین داده های ماتریس ورودی را برای بخش های دو در دو ماتریس خواهیم داشت.
* نکته اینکه در این فرآیند ماتریس های کوچک تر که از ماتریس اصلی در نظر گرفته می شوند با همدیگر همپوشانی (اورلپ) ندارند و لذا ماتریس خروجی ماتریس کوچک تری خواهد بود. مشابه تصویر زیر.
* به این فرآیند Average(mean) Pooling گفته می شود
* از این روش برای کاهش ابعاد ماتریس و خصوصیات در مدل های یادگیری عمیق استفاده می شود..
* استفاده از اوریج پولینگ در هموار سازی تصاویر استفاده می شود. خصوصا تصاویر نویزی که پیکسل های نامرتبط دارند مثل عکس های آسیب دیده.



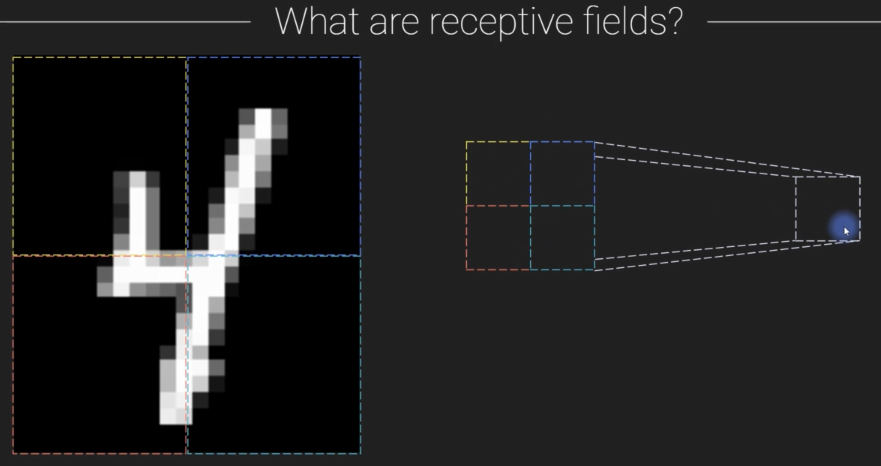
**Max Pooling**

* زمانیکه بخواهیم داده های ماکزیموم را از ماتریس های داده شده کوچک تر (قسمت های دو در دو) که در دل ماتریس داده های ورودی هستند جداسازی کنیم و در یک ماتریس کوچک تر ذخیره کنیم باید از روش مکس پولینگ استفاده نماییم.
* استفاده از مکس پولینگ در مشخص نمودن داده های با تفاوت های زیاد (کانتراست تصویر) استفاده می شود و می توان با این روش کنتراست را زیاد کرد.
* پیدا کردن لبه ها یکی از کاربرد های مکس پولینگ است.
* استفاده از اوریج پولینگ در هموار سازی تصاویر استفاده می شود. خصوصا تصاویر نویزی که پیکسل های نامرتبط دارند مثل عکس های آسیب دیده.

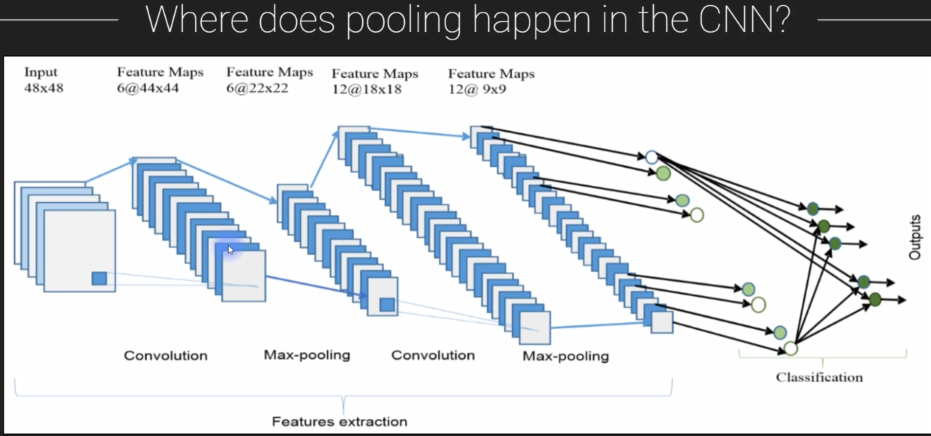


**فیلدهای پذیرا (Receptive Fields)**

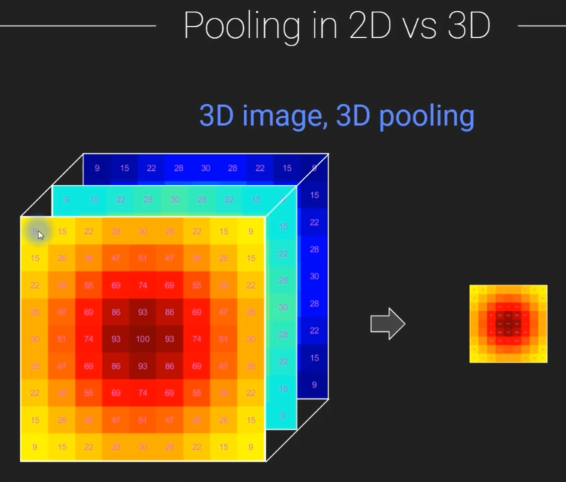
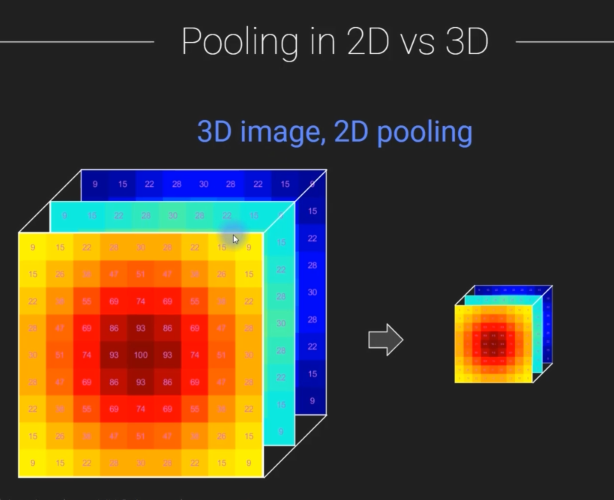
* مفهوم فیلد های پذیرا این است که وقتی از روش های مکس یا مین پولینگ برای کاهش بعد تصاویر استفاده می کنیم، بطور مثال هر چهار سلول ماتریس داده های ورودی به یک سلول تبدیل می شود. یک سلول خروجی (نتیجه مکس پولینگ) بعنوان فیلد پذیرای تصویر ورودی نامیده می شود.



* استفاده از کانولوشن و مکس پولینگ در مدل های سی ان ان بصورت جفتی و در امتداد همدیگر مرسوم است.



* در پولینگ تصاویر با سه لایه رنگی (RGB) می توان از دو روش استفاده نمود.
  + تبدیل تصویر با همان لایه های سه گانه : بدین معنی که هر لایه بصورت جداگانه پولینگ شود و خروجی ماتریس های سه گانه کوچک تر باشند.
  + تبدیل تصویر از لایه های سه گانه به یک لایه : در این حالت ماتریس های بطور مثال دو در دو از لایه قرمز و سبز و آبی که در امتداد هم هستند در نظر گرفته می شود و از 12 سلولی که مشخص شده است فرآیند پولینگ انجام می شود و در نهایت یک عدد خروجی بدست می آید.



**مقایسه بین پولینگ و استراید (Pooling vs Stride)**

* هر دو روش برای کاهش ابعاد تصویر استفاده می شوند.
* روش کار استراید به این صورت است که از روی تعداد مشخص شده (توسط کاربر) می پرد و در ماتریس ضرایب کرنل ضرب نمی کند.
* روش کار پولینگ این است که تعداد مشخص شده (توسط کاربر) ماتریسی را در ماتریس داده های اصلی جدا و ماکزیموم یا مینی موم یا میانگین آنها را بعنوان یک عدد بر می گرداند.

|  |  |
| --- | --- |
| **پولینگ** | **استراید** |
| از نظر محاسباتی سریعتر است | از نظر محاسباتی کمی کند تر است |
| پارامتری ندارد | دارای پارامتر های یادگیرنده است |
| کرنل کوچکتری دارد | کرنل نسبتا بزرگتری دارد |
| خیلی با ثبات است | ممکن است در مدل های پیچیده دچار اختلال شود |

* استفاده از این روش ها در مدل های یادگیرنده عمیق سلیقه ای است و کسی نمی تواند بگوید کدام بهتر است. شاید بهتر باشد که هردو را تست کنیم. روش پولینگ دارای سابقه استفاده بیشتری بوده است.

**تبدیل تصاویر (Image Transforms)**

* دو دلیل عمده برای تبدیل تصاویر وجود دارد:
  + مدل های سی ان ان از قبل آموزش دیده (Pre-trained CNNs) با تصاویری با ابعاد مشخص آموزش دیده اند و برای استفاده از آنها باید ابعاد تصاویر تغییر کند ویا بصورت سیاه و سفید به مدل ارائه شود.
  + با تبدیل تصاویر می توان داده های بیشتری برای آموزش مدل بدست آورد. در واقع با تبدیل تصاویر داده های کم ارزش تر کنار گذاشته می شوند و داده های با ارزش تر که محتوای تصاویر را حفظ می کنند باقی می مانند. یکی از روش های تبدیل تصاویر data augmentation است مثل تغییر رنگ، تغییر اندازه، تغییر جهت و ...

**تابع ToTensor در torchvision.transforms**

* از تابع ToTensor() برای تبدیل تصاویر یا آرایه های نامپای به تنسور استفاده می شود.
* استفاده از این تابع علاوه بر تبدیل ماتریس ها به تنسور، مقیاس داده ها را نیز انجام می دهد و از بازه صفر تا 255 به بازه صفر و یک تبدیل می کند.
* علاوه بر آن در آرایه ها ساختار داده ها بصورت (H x W x C) یعنی (کانال رنگ x عرض x ارتفاع ) می باشد که در تنسور باید بصورت (C x H x W ) تبدیل شود و این تابع آن را انجام می دهد.