گزارش پروژه

نوید کنعانی
۹۸۱۱۱۰۳۳۳۰۲۰۲۷
imnavidkanaani@gmail.com

کاردانی - نرم افزار

درس مباحث ویژه برنامه نویسی

۲۴ دی ۱۳۹۹

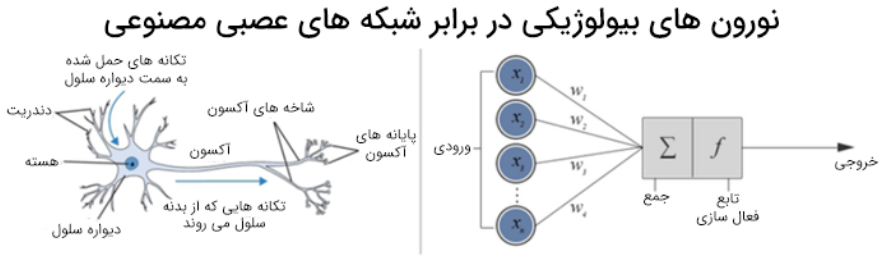
شرح پروژه

- موضوع: طبقه بندی تصاویر به کمک یادگیری ماشین
- ابزارها :
  Pyhton, Pandas, Numpy, MatPlotlib, TensorFlow, Google Colab
- منبع داده : Dog Breed Identification from Kaggle Competition
- الگوریتم : MobileNet_V2

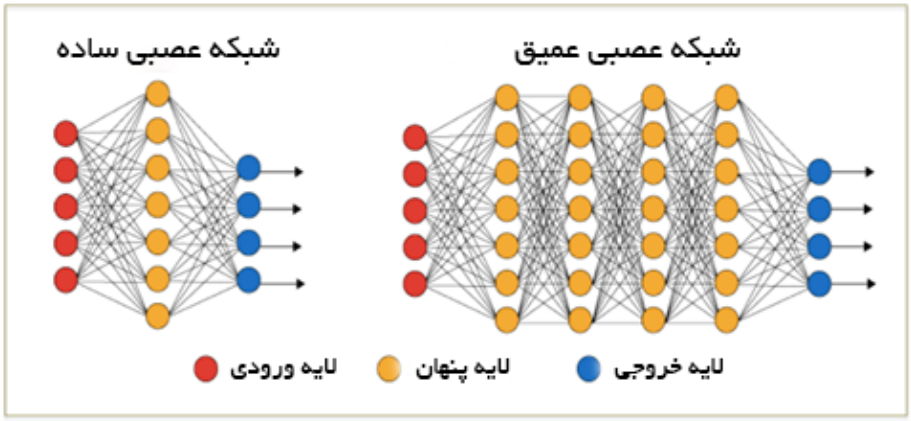
مقدمه

در این پروژه هدف پیاده سازی پردازش تصویر و طبقه بندی تصویر با استفاده از یادگیری عمیق و یادگیری تحت نظارت و Transfer Learning می باشد. استفاده از تکنیک Transfer Learning به این دلیل بوده که ما گاهی اوقات برای حل مسائل مختلف می توانیم از الگوریتم هایی استفاده کنیم که قبلا آموزش دیده اند و با اینکار در منابع و زمان صرفه جویی می کنیم و همچنین دیگر درگیر جزییات الگوریتم ها نمی شویم و مسئله را حل می کنیم.
قبل از هر چیز مروری بر یادگیری عمیق و مفاهیم پایه ای و نحوه پردازش تصویر در الگوریتم شبکه های عصبی تکاملی داشته باشیم. یادگیری عمیق یکی از تکنیک های آماری یادگیری ماشین می باشد. روند یادگیری و استخراج ویژگی از داده ها توسط الگوریتم شبکه های عصبی مصنوعی انجام می شود. ساختار شبکه های عصبی مصنوعی از ساختار مغز انسان ها و موجودات زنده الهام گرفته شده است. به این صورت که ما یک شبکه عصبی بیولوژیکی داریم که به سیستم عصبی ما متصل است. مغز ما شبکه بسیار پیچیده ای است که تقریبا از ۱۰ میلیارد نرون تشکیل شده که هر کدام از آنها به ۱۰ هزار نرون دیگر متصل هستند. این نرون ها هر کدام سیگنالی را به عنوان ورودی دریافت می کنند و در صورتی که آن از حد آسنانه آن نرون بالاتر برود باعث خارج شدن سیگنال می شود و سیگنال به نرون بعدی منتقل می شود، در نتیجه آن باعث بوجود آمدن واکنش های ما می

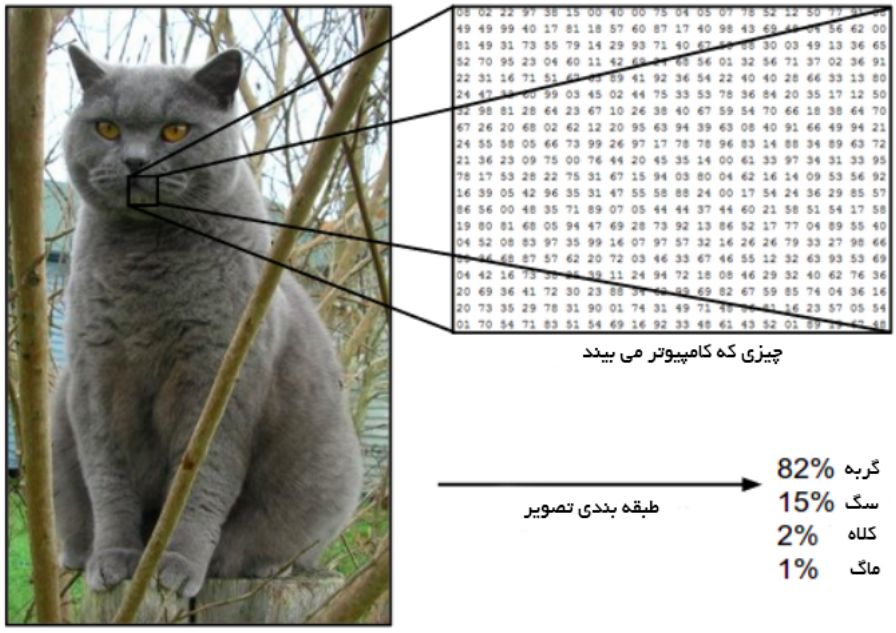
شوند.



شبکه های عصبی مصنوعی هم از همین اصل بیروی می کنند و داده هایی که ما به عنوان ورودی به الگوریتم می دهیم در نود های ورودی تقسیم می شوند و پس از انجام محاسبات و اعمال وزن ها در صورتی که از حد آستانه آن گره بالاتر رود خروجی تولید می شود؛ که این خروجی به عنوان ورودی نود بعدی استفاده می شود. شبکه های عصبی مصنوعی از دو لایه خروجی و ورودی تشکیل شده اند (لایه ورودی را معمولا به شمار نمی آورند) و عمیق بودن یادگیری به تعداد لایه های پنهان آن اشاره می کند.



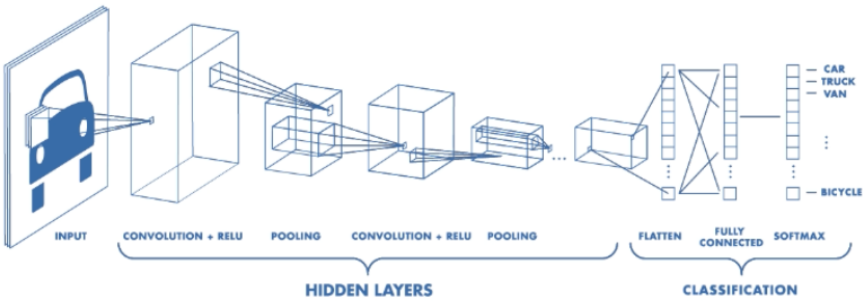
حال یک روش برای پردازش تصویر در یادگیری عمیق استفاده از شبکه های عصبی تکاملی است. در شبکه هر تصویر به صورت یک آرایه سه بعدی از اعداد وجود دارد که هر کدام به عنوان پیکسل شناخته می شوند و دارای طول، عرض و عمق می باشند. طول و عرض به رزولویشن تصویر بستگی دارد و عمق همان کانال رنگی (قرمز، سبز، آبی) است.



به طور کلی در این شبکه تصویر به عنوان ورودی داده می شود و از لایه های پشت هم یا فیلتر ها رد می شود تا ویژگی ها آنرا استخراج کند. نکته ای که وجود دارد این است که در ابتدا الگوریتم ما و فیلترها نمی دانند که در کجای تصویر به دنبال ویژگی هایی مانند لبه ها یا منحنی ها باشند و وزن هایی که قبلا گفته شد، اعدادی تصادفی هستند، درست شبیه به ذهن یک کودک که با همه چیز نا آشناست. معمولا ما یک مجموعه داده آموزشی بزرگ از هزاران تصویر همراه برچسب های از پیش تعیین شده داریم. مدل در ابتدا یک مرحله جلو می رود، وزن های اولیه را محاسبه می کند و برچسب خروجی را پیش بینی می کند (برای مثال یک سگ است) و آنرا با مقدار درست آن که در داده های آموزشی ما می باشد مقایسه می کند. از آنجا که داده های ما آموزشی بوده و ما نتیجه برچسب ها را از قبل می دانستیم. باتوجه به موفق بودن یا نبودن پیش بینی اصطلاحا یک تابع از دست رفته [1] محاسبه می شود. سپس شبکه به عقب باز می گردد تا وزن هایش را تنظیم کند. روشی که کامپیوتر برای تنظیم وزن هایش به منظور کاهش

[1] Loss Function

تلفات انجام می دهد، تابع پَس انتشار [۲] گفته می شود. حالا مدل در شبکه یک مرحله به عقب می رود تا وزنی که بیشترین تلفات را داشته تعیین کند و همچنین بهترین وزن با کمترین تلفات را پیدا کند. انتظار می رود که در شروع کار میزان تلفات حساب شده بالا باشد و با مقداری گذر به عقب و جلو به حداقل مقدار خودش می رسد. در نهایت شبکه ما به خوبی آموزش داده شده و وزن هایش را به درستی تنظیم کرده است.



شرح مسئله و توضیحات الگوریتم

فرض کنید که در خیابان در حال راه رفتن هستید و سگی را می بینید و کنجکاو می شوید که نژاد آن سگ چیست. اولین گزینه ای که به ذهن ما می رسد این است که کاش برنامه ای بود که از روی تصاویر سگ ها نژاد آنها را تشخیص می داد. خب این مسئله براحتی با کمک یادگیری ماشین و هوش مصنوعی قابل حل است. اما داده های ما ازنوع تصویر هستند و بدون ساختارند؛ بنابراین باید از تکنیک یادگیری عمیق و پردازش تصویر که در بالا توضیح دادیم، استفاده کنیم. همچنین می دانیم که الگوریتم های یادگیری عمیق پیچیدگی بالایی دارند و منابع محاسباتی زیادی نیاز دارند و باید الگوریتم ما به اندازه کافی بهینه باشد که روی موبایل اجرا شود.

---

[۲] Back prropagation

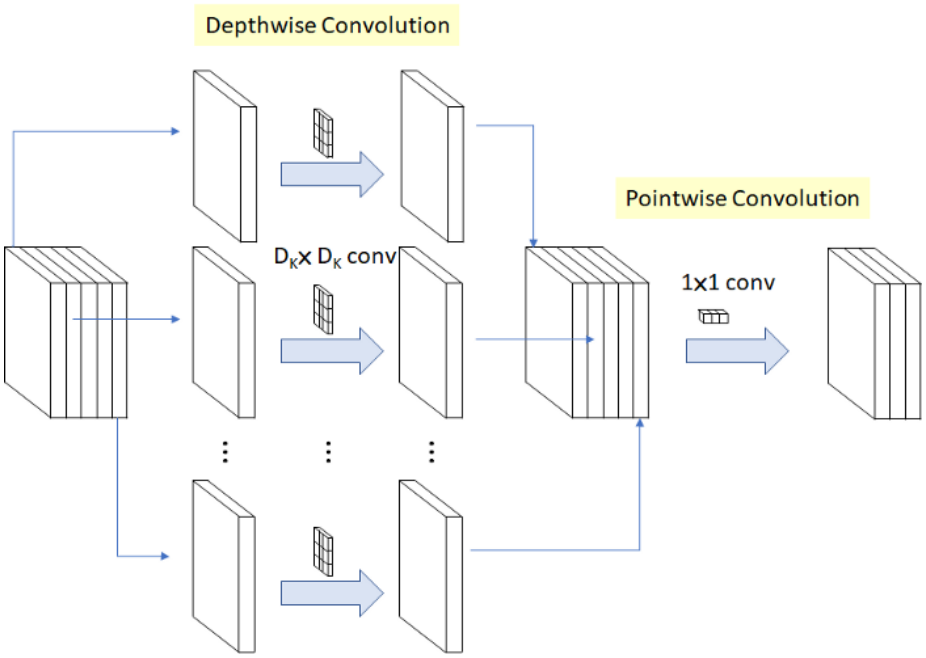
MobileNet-V1

الگوریتمی که ما در اینجا استفاده کردیم به نام MobileNetV2 است که نسخه بهینه شده MobileNetV1 است و توسط گوگل ارائه شده است. این الگوریتم پیچدگی زیادی ندارد و برای اجرا بروی دستگاه های موبایل و پشتیبانی مسائل طبقه بندی و تشخیص، بوجود آمده است. برای کاهش اندازه و پیچیدگی مدل از کانولوشن های جداشونده عمیق استفاده می کند. زمانی که تعداد پارامتر ها کم تر باشد اندازه مدل کوچک تر است و زمانی که تعداد جمع و ضرب کم تر باشد پیچیدگی آن کم تر است. بنابراین الگوریتم با دو پارامتر ضریب عرض α و ضریب رزولویشن ρ خود را تنظیم می کند.

Depthwise Separable Convolution

Depthwise Convolution

کانولوشن جداشونده عمیق، ترکیب های عمیقی هستند که با یک کانولوشن نقطه ای همراه می شوند. مانند تصویر زیر:



کانولوشن عمیق، کانولوشن فاصله ای با کانال DK x DK است. مثلا در تصویر بالا ما ۵ کانال داریم، بنابراین ۵ کانولوشن DK x DK فاصله دار داریم.

Pointwise Convolution

در واقع همان کانولوشن ۱×۱ برای تغییر بُعد است.
باتوجه به گفته های بالا تابع Cost :

$$D_K.D_K.M.D_F + M.N.D_F.D_F$$

M : تعداد کانال های ورودی است. N: تعداد کانال های خروجی است. D_K: اندازه هسته است. و D_F اندازه ترکیب ویژگی هاست.
در کانولوشن استاندارد به صورت:

$$D_K.D_K.M.N.D_F.D_f$$

است که در نتیجه کاهش محاسبات:

$$\frac{D_K.D_K.M.D_F+M.N.D_F.D_F}{D_K.D_K.M.N.D_F.D_f} = \frac{1}{n} + \frac{1}{D_k^2}$$

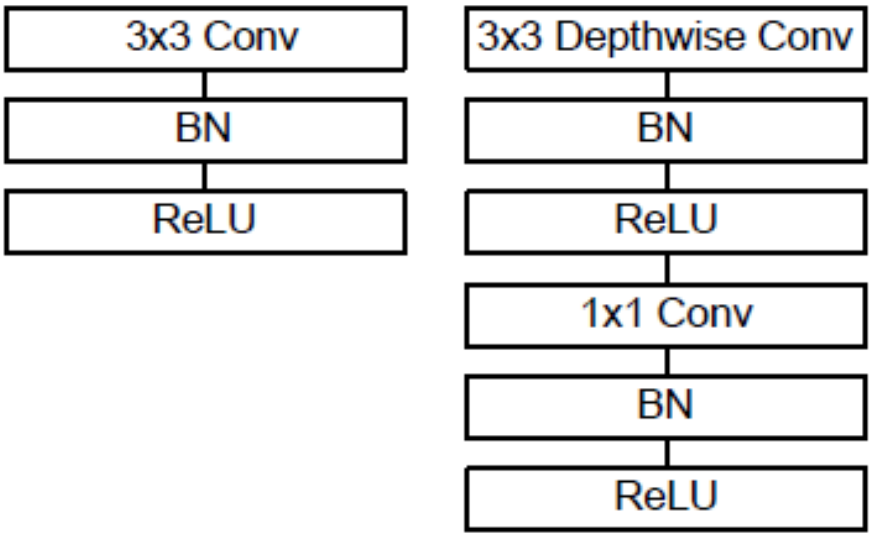
زمانی که DK x DK برابر ۳×۳ باشد، ۸ تا ۹ برابر محاسبات کم تری نیاز دارد و البته دقت کمی کاهش میابد.

معماری کلی شبکه

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

بعد از هر کانولوشن Batch Normalization (BN) و ReLU اعمال می شوند.



در تصویر زیر کانولوشن استاندارد و کانولوشن جداشونده عمیق مقایسه شده اند.

Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

مشاهده می کنیم که موبایل نت تنها با ۱ درصد کاهش دقت ولی با کاهش چشم گیر پارامترها و جمع و ضرب ها عمل کرد.

ضرب کننده عرض α برای مدل کوچک تر

ضرب کننده عرض α برای کنترل عرض ورودی های یک لایه تعریف می شود، که دراین صورت M به ۸M تبدیل می شود. و Cost کانولوشن جداشونده عمیق می شود:

$$D_K.D_K.\alpha M.D_F.D_F + \alpha M.\alpha N.D_F.D_F$$

که در آن α بین ۰ تا ۱ است معمولا ۰.۲۵، ۰.۵، ۰.۷۵ و ۱ تنظیم می شود. مقدار ۱ برای α مقدار پیشفرض موبایل نت است. و مقدار محاسبات و تعداد پارامترها به صورت مربع یا درجه دو با α کاهش میابد.

Table 6. MobileNet Width Multiplier

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

همانطور که مشاهده می کنید، دقت به آرامی با از آلفا ۱ تا ۰.۲۵ که کوچک تر است، کاهش میابد.

ضرب کننده رزولویشن ρ برای کاهش نماینده ها

ضرب کننده وضوح ρ برای کنترل رزولوشن تصویر ورودی شبکه تعریف می شود. با این ضرب کننده تابع Cost می شود:

$$D_K.D_K.\alpha M.\rho D_F.\rho D_F + \alpha M.\alpha N.\rho D_F.\rho D_F$$

که در آن ρ بین ۰ تا ۱ است. و رزولوشن ورودی ۲۲۴، ۱۹۲، ۱۶۰ و ۱۲۸ هستند. مقدار پیشفرض موبایل نت $\rho = 1$ است.

Table 7. MobileNet Resolution

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

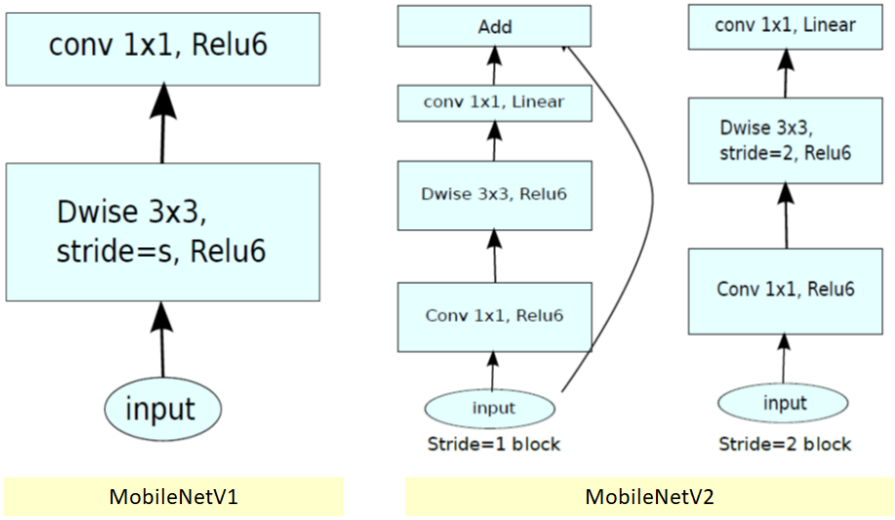
همانطور که مشاهده می کنید با پایین آمدن رزولوشن، ذقت هم کاهش میابد.

MobileNet-V2

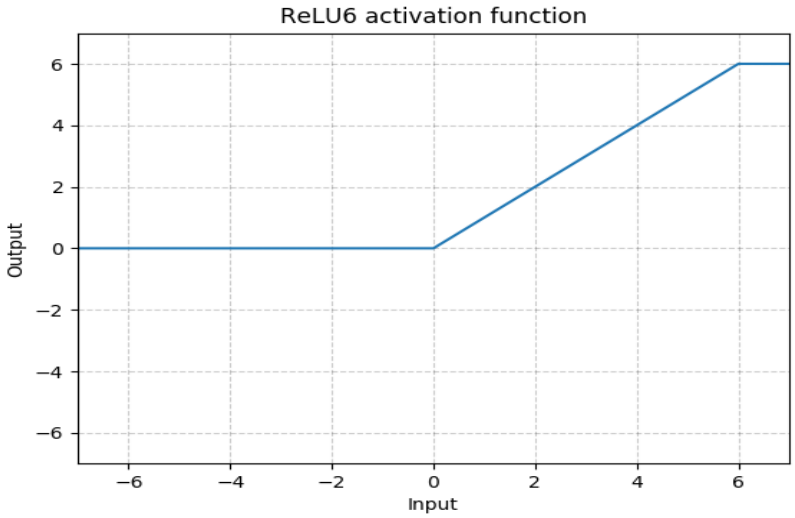
توضیحاتی که داده شد در رابطه با MobileNet-V1 بود و نسخه دوم آن که ما از آن استفاده می کنیم، بهنه تر شده و تغییراتی داشته است. این الگوریتم یک ساختار باقیمانده معکوس معرفی می کند و غیرخطی بودن لایه های باریک در این نسخه ازبین رفته است. این الگوریتم به عنوان ستون فقرات برای استخراج ویژگی، عملکرد بروز و همچنین تشیخیص آبجکت [۳] و تقسیم بندی معنایی [۴] شناخته می شود.

[۳] Object Detection
[۴] Semantic Segmentation

بلوک های MobileNet-V2



همانطور که در تصویر مشاهده می کنید. در نسخه اول موبایل نِت دو لایه وجود دارد. لایه اول که کانولوشن عمقی نامیده می شود. با اعمال یک فیلتر کانولوشنالی برای ورودی هر کانال، فیلترینگ سبک را انجام می دهد. لایه دوم که یک کانولوشن ۱×۱ است. کانولوشن نقطه ای نام دارد و وظیفه اش ساخت ویژگی های جدید با محاسبه ترکیبات خطی ورودی کانال ها می باشد.



ReLU6 در اینجا برای مقایسه استفاده می شود. دلیل استفاده از آن هم توانمندی اش در محاسبات با دقت پایین است.

در MobileNet-V2 دو نمونه بلوک وجود دارد. یک بلوک باقی مانده با گام ۱ و بلوک دیگر با گام ۲ برای کوچک سازی. برای هر دو نوع بلوک ۳ لایه وجود دارد. در اینجا اولین لایه کانولوشن ۱×۱ همراه ReLU6 است. لایه دوم کانولوشن عمقی است. و لایه سوم کانولوشن ۱×۱ دیگری است اما هیچ تابع غیر خطی ندارد.

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

و در اینجا یک فاکتور انبساط t است. که برای تمام آزمایشات اصلی مقدار ۶ را دارد. اگر ورودی ۶۴ کانال داشته باشد، خروجی داخلی $64 \times t = 64 \times 6 = 384$ کانال خواهد داشت.

معماری نهایی

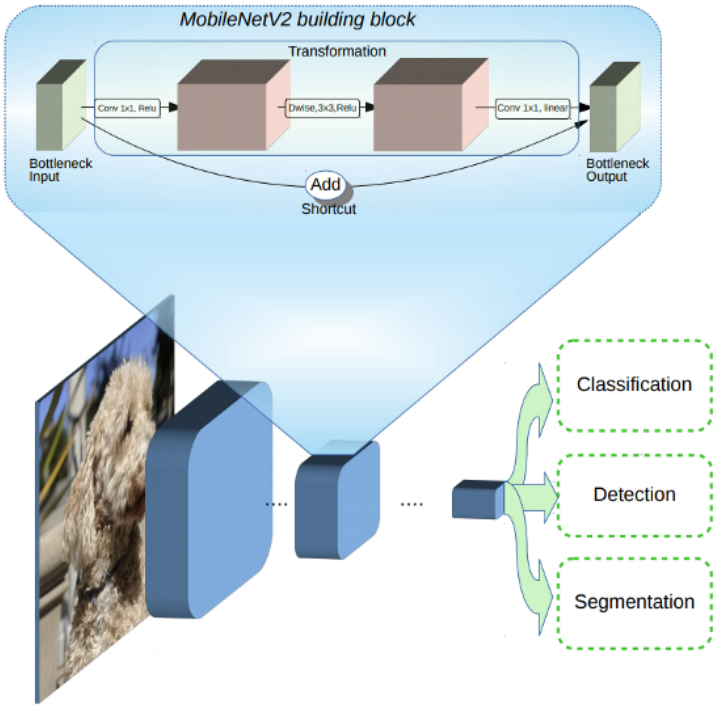
Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

که در آن t : فاکتور انبساط یا بسط، c : تعداد کانال های خروجی، n : تعداد

تکرار، s : گام است. و هسته های ۳×۳ برای کانولوشن فاصله دار استفاده می شود. به طور معمول شبکه اصلی (ضرب کننده عرض ۱، ۲۴۴×۲۴۴) میزان محاسبات ۳۰۰ میلیون جمع و ضرب را دارد و ۴.۳ میلیون پارامتر استفاده می کند. (ضرب کننده عرض در قمست MobileNet-V1 توضیح داده شد). عملکرد سبک و سنگین کردن بیشتر مورد بررسی قرار می گیرد و برای رزولوشن ورودی از ۹۶ تا ۲۲۴ و ضرب کننده عرض از ۳۵.۰ تا ۴.۱ در نظر می گیرد. میزان محاسبات شبکه تا ۵۸۵ میلیون جمع و ضرب بالا می رود و این درحالیست که اندازه مدل بین ۷.۱ میلیون و ۹.۶ میلیون پارامتر متغیر است. برای آموزش شبکه ۱۶ واحد پردازنده گرافیکی (GPU) با بچ های به اندازه ۹۶ استفاده می شود.

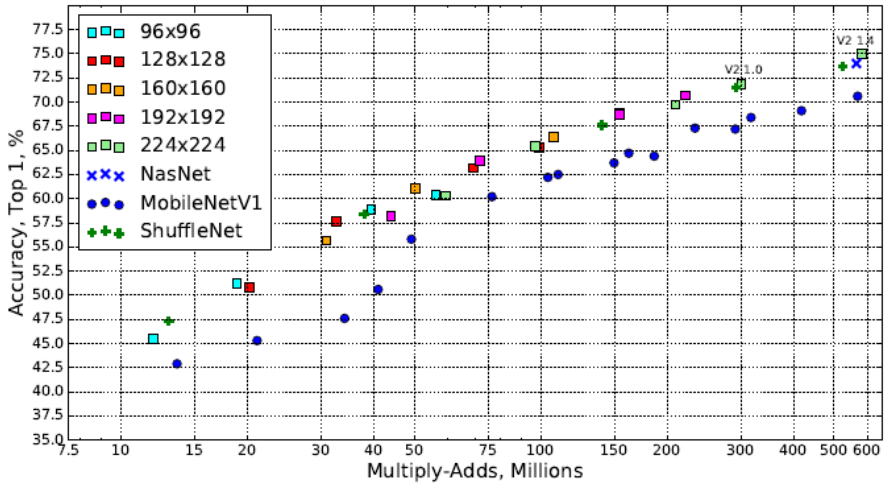
Size	MobileNetV1	MobileNetV2	ShuffleNet (2x,g=3)
112x112	64/1600	16/400	32/800
56x56	128/800	32/200	48/300
28x28	256/400	64/100	400/600K
14x14	512/200	160/62	800/310
7x7	1024/199	320/32	1600/156
1x1	1024/2	1280/2	1600/3
max	1600K	**400K**	600K

مقایسه نتایج



MobileNet-V2 عملکرد بهتری نسبت به MobileNet-V1 و ShuffleNet(1.5) از نظر اندازه مدل و هزینه محاسبات داشته است. با ضرب کننده عرض ۴.۱، MobileNet-V2 عمل کرد بهتری نسب به ShuffleNet(x2) و زمان استنباط سریعتری نسبت به NasNet داشته است.

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**



همانطور که در نمودار بالا مشاهده می کنید، با استفاده از ورودی رزولوشن ها و ضرب کننده عرض متفاوت، همواره نسحه دوم عملکرد بهتری نسبت به نسخه اول داشته است.

توضیحات مربوط به پروژه

خب بعد از صحبت درباره مطالب فنی و جزییات الگوریتم در ادامه دربازه روند انجام پروژه و مراحل مختلف آن صحبت می شود.

مسئله ما در اینجا ایجاد برنامه ای است که به کمک آن بتوانیم نژاد سگ ها را از روی تصاویر آنها طبقه بندی کنیم. برای اجرای آن نیاز به یادگیری ماشین داریم و از آنجایی که داده های ما از نوع تصویر می باشند و بدون ساختار هستند، از یادگیری عمیق باید استفاده کنیم. دیتای ورودی ما برچسب گذاری شده است و برچسب دیتای خروجی را هم می دانیم، بنابراین یادگیری ما تحت نظارت است.

مراحل انجام پروژه به شرح زیر است:

۱. جمع آوری و مرتب کردن دیتا (تبدیل داده ها به Tenors)

۲. انتخاب مدل

۳. وارد کردن دیتای ورودی به مدل و پیش بینی کردن

۴. ارزیابی مدل

۵. بهبود مدل

۶. ذخیره مدل

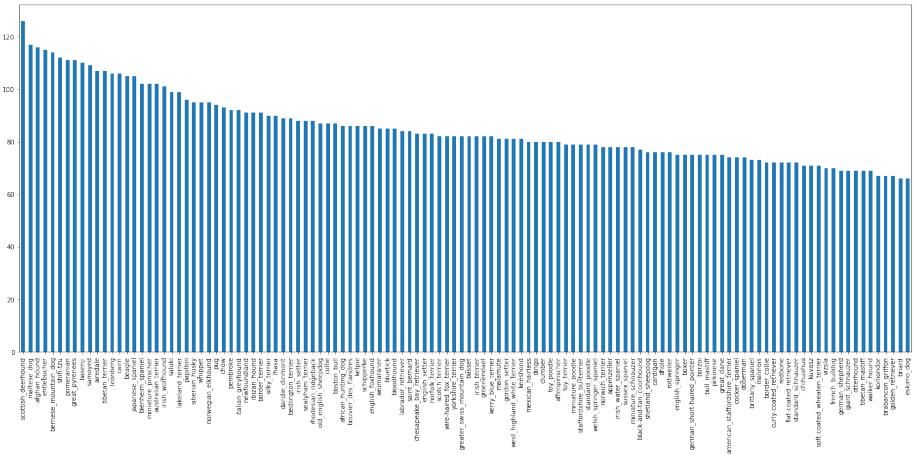
۱ جمع آوری و مرتب کردن داده ها

در مسائل یادگیری ماشین مهم ترین بخش مربوط به داده های ما می باشد و مقدار و نوع آنها مهم هستند. داده هایی که در این مسئله جمع آوری شده طبق مسئله از نوع تصویر هستند. بنابراین بدون ساختار می باشند. از طرفی در یادگیری ماشین نیاز است که دیتا به صورت مرتب شده و آماده برای مدل باشد. بنابراین ما باید عملیاتی روی دیتایمان انجام دهیم و آنها مرتب کنیم چرا که در صورت داشتن دیتای نامناسب، یادگیری مدل درست انجام نمی شود و دچار خطا در نتیجه نهایی می شود یا سرریز می کند. مرتب کردن داده بر اساس نوع آنها متفاوت می باشد. در این پروژه داده های ما بدون ساختارند و از نوع تصویر هستند و ما از یادگیری عمیق می خواهیم استفاده کنیم. اولین کاری که باید انجام دهیم تعداد داده ها و برچسب ها را بررسی کنیم و در صورتی که برابر نبودند آنها را اضافه کنیم یا کاهش دهیم. به طور کلی کامپیوتر با اعداد سروکار دارد و داده ها در فرمی که برای ما قابل مشاهد است، برای کامپیوتر ها قابل درک نیست. در فریمورک TensorFlow داده ها باید به صورت Tensor باشند و مدل ما آنها را می شناسد. بنابراین ما باید داده هایمان را به مقادیر عددی (Tensors) تبدیل کنیم.
جزییات دیتای آموزشی ما در این پروژه به شرح زیر است:

```
                                    id                breed
count                            10222                10222
unique                           10222                  120
top     a4263bf0c3841ddd5ed4b0c40cbd6cf9  scottish_deerhound
freq                                 1                  126
```

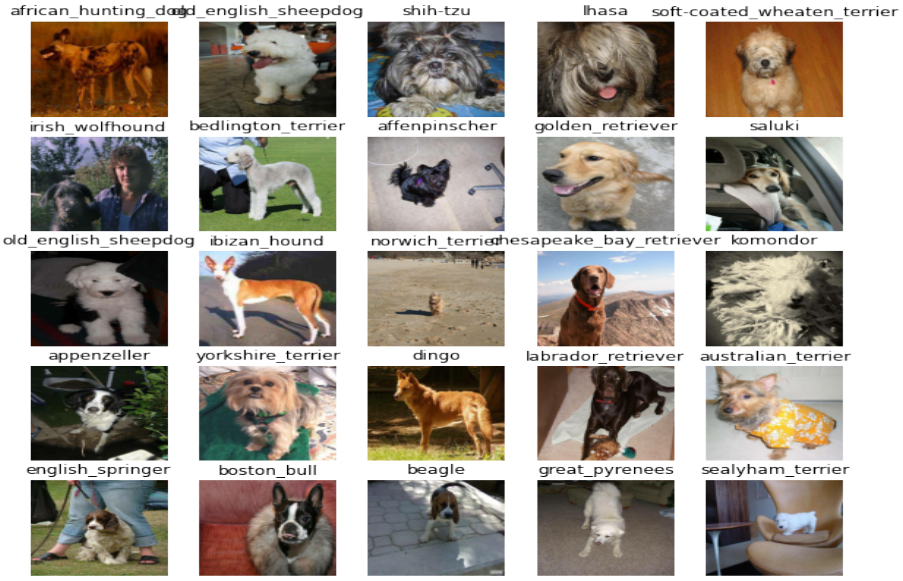
	id	breed
0	000bec180eb18c7604dcecc8fe0dba07	boston_bull
1	001513dfcb2ffafc82cccf4d8bbaba97	dingo
2	001cdf01b096e06d78e9e5112d419397	pekinese
3	00214f311d5d2247d5dfe4fe24b2303d	bluetick
4	0021f9ceb3235effd7fcde7f7538ed62	golden_retriever

همانطور که در تصویر مشخص است ما مقدار ۱۰۲۲۲ نمونه داده داریم و تعداد ۱۲۰ نژاد (برچسب) به طور کلی که هر کدام از نمونه ها یک نوع نژاد را دارند. تعداد کل برچسب ها با تعداد کل نمونه ها هم برابر می باشند.



در نمودار بالا تعداد نمونه از هر نمونه به تصویر کشیده شده است که به طور میانگین ۸۲ تصویر برای هر نژاد (برچسب) وجود دارد. فیلد id که در تصویر مشخص است همان نام فایل های تصاویر می باشند. بنابراین ما نیاز به یک لیستی از آدرس و نام فایل ها داریم، لیستی از برچسب ها، لیستی از مقدار یکتای نژادها (برچسب ها) داریم.

در داده هایی که جمع آوری شده اند ما مجموعه داده ارزیابی [۵] نداریم و خودمان آنرا ایجاد می کنیم. پس داده های آموزشی را تقسیم می کنیم و بخشی از آنرا (۲۰٪) برای ارزیابی مدل در نظر می گیریم. بعد از انجام این مراحل و آشنا شدن با دیتا، باید آنها را پردازش کنیم و به Tensors تبدیل کنیم. در ادامه مدل دیتا را به صورت دسته [۶] شده به عنوان ورودی می گیرد و خروجی می دهد. اندازه این دسته ها مشخص است و در اینجا ما ۳۲ در نظر گرفتیم که مقدار پیشفرض آن در TensorFlow می باشد. همچنین نکته مهم این است که تصاویر احتمالا در اندازه های متفاوت می باشند و این موضوع باعث ایجاد تداخل در یادگیری مدل می شود و ما باید اندازه تمام آنها را به یک اندازه تغییر دهیم. بنابراین اندازه آنها را به ۲۲۴ تغییر می دهیم که مقداری است که در داکیومنت های مدل ذکر شده است. در ادامه خواهیم دید که چرا اندازه تصاویر باید یکی باشند. بعد از انجام این مراحل تصاویر ما در قالب Tensors به آرایه های سه بعدی (طول، عرض، عمق) از پیکسل ها تبدیل می شوند. منظور از عمق که بعد سوم می باشد کانال رنگی تصاویر هستند که در اینجا ۳ به معنای قرمز، سبز و آبی است. در آخر داده های آموزشی و آزمایشی به صورت زیر از دید ما در می آیند.



[۵] Validation dataset
[۶] Batch

۲ مراحل ۲ به بعد

بعد از مرتب کردن و آماده کردن دیتا نوبت به انتخاب مدل مناسب و آماده کردن مدل می رسد. مدل های مختلفی برای پردازش تصویر وجود دارند. مدلی که من انتخاب کردم MobileNet_v2 است. این مدل سبک است و برای طبقه بندی رو دیوایس های مختلف با استفاده از معماری شبکه های عصبی استفاده می شود. همانطور که قبلا هم گفته شد ما از تکنیک Transfer Learning استفاده کردیم و این به این معنای استفاده از مدل های مشترک برای مسائل مشابه هم می باشد. این مدل قبلا توسط پایگاه داده عظیم ImageNet [۷] آموزش دیده است. این مدل تصاویر را در اندازه ۲۲۴*۲۲۴ به عنوان ورودی می گیرد و دلیل اینکه ما تصاویر را به این اندازه تغییر دادیم همین موضوع بود.

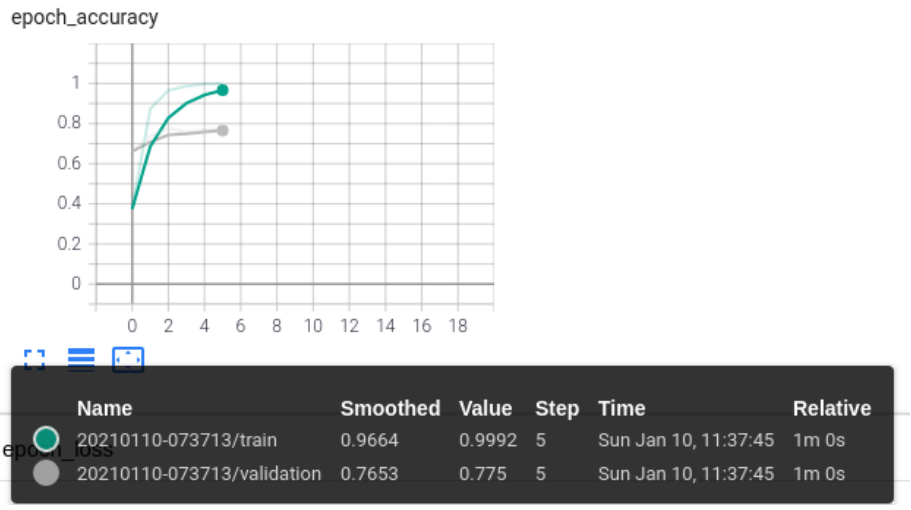
در پردازش تصویر الگوریتم تصاویر را قطعه قطعه می کند و ابتدا لبه های تصویر را پیدا می کند تا بفهمد که در کجای تصویر باید به دنبال ویژگی های آن باشد. البته جزییات آن خیلی بیشتر از این تعریف است اما به طورکلی به این صورت عمل می کنند. قبل از ایجاد مدل فرم داده های ورودی و خروجی را مشخص می کنیم. به دلیل استفاده از شبکه های عصبی الگوریتم از لایه ها متفاوت تشکیل شده است و باید آنها را تنظیم کنیم و فرم داده های ورودی و خروجی را به الگوریتم بدهیم. مدل را ایحاد می کنیم.

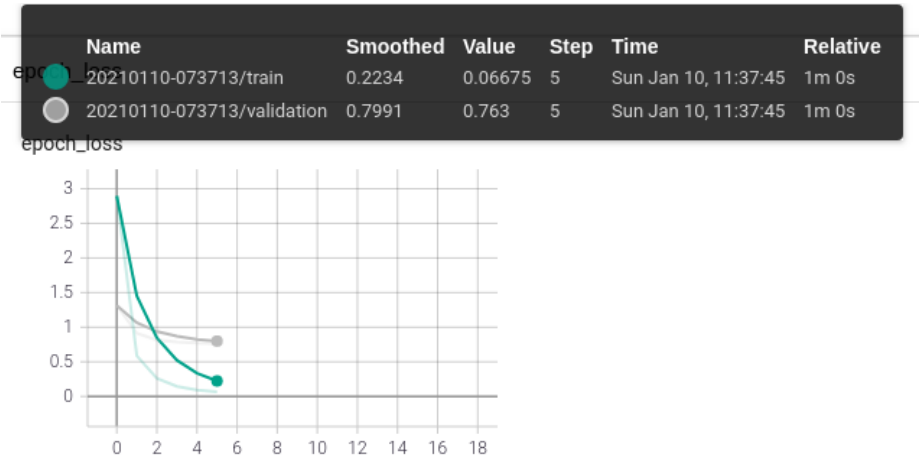
```
Building model with:  https://tfhub.dev/google/imagenet/mobilenet_v2_130_224/classification/4
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     (None, 1001)              5432713
_________________________________________________________________
dense (Dense)                (None, 120)               120240
=================================================================
Total params: 5,552,953
Trainable params: 120,240
Non-trainable params: 5,432,713
_________________________________________________________________
```

در تصویر بالا sequential به این معناست که طبقه بندی ما چندگانه [۸] است. لایه ها مدل مشخص شده اند و در مجموعه ۵۵۲۹۵۳ پارامتر (ویژگی) الگوریتم پیدا کرد که ۱۲۰۲۴۰ آنها را برای یادگیری انتخاب کرده است. در روند یادگیری

[۷] ImageNet
[۸] Multi-class Classification

ما از callbacks استفاده می کنیم. که این ابزار ها به یادگیری مدل ما کمک می کنند و کاربردهای متفاوتی دارند. برای مثال یکی از آنها که ما از آن در برنامه استفاده کردیم Early Stopping است. این ابزار روند یادگیری را کنترل می کند و در صورت Overfitting متوقف می کند. یا نمونه دیگر که در برنامه استفاده شده است TensorBoard می باشد که قابلیت به تصویر کشیدن در تنسرفلو را به ما می دهد. همچنین از بهنیه ساز ها و محاسبه کننده توابع از دست رفته نیز استفاده شده است. بعد از تنظیم و ایجاد مدل، پیشنهاد می شود مدل را با بخشی از داده ها آموزش بدهیم تا از کارکرد درست آن مطمئن شویم. برای مثال من ۳۰۰۰ نمونه که ۶۰۰ نمونه آزمایش و ۲۴۰۰ نمونه آموزشی می باشند، مدل را تغذیه کردم.



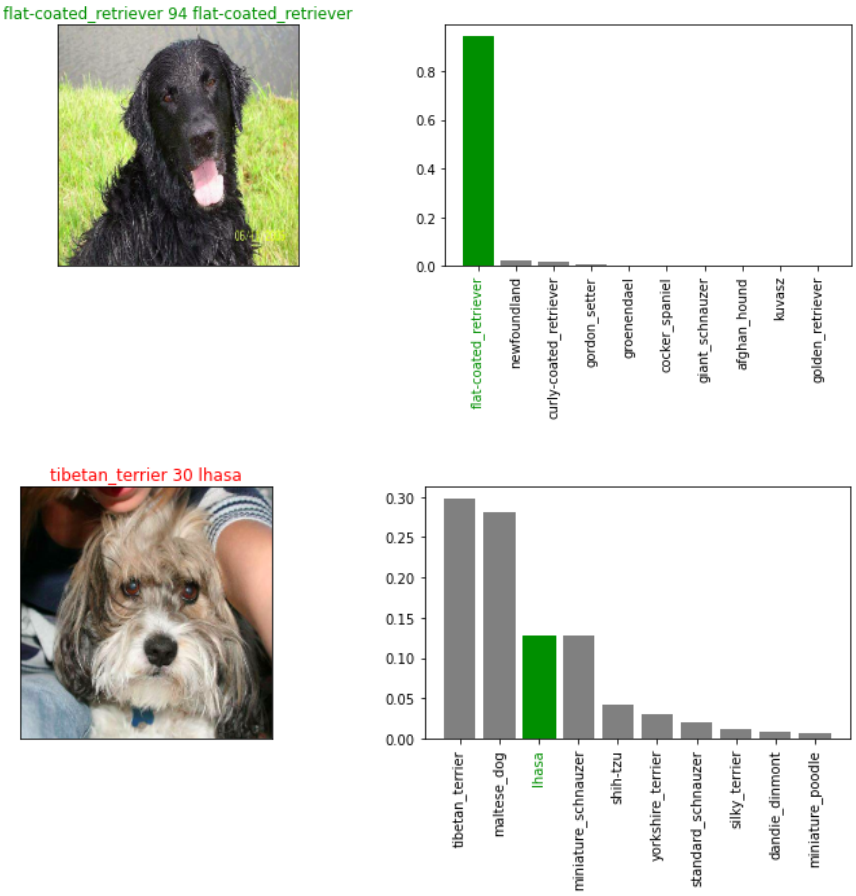


همانطور که در تصاویر مشاهده می کنید دقت مدل رشد داشته و خطای آن نزول داشته که این به این معناست که مدل ما یادگرفته و آموزش موفق بوده است. نکته: اگر دیتای نمونه کم تری به مدل تغذیه کنیم ممکن است Over-Fitting رخ دهد که در بار اول طبیعی است به دلیل دیتای کم بوده و باز هم به این معناست که مدل یادگرفته است و مشکلی ندارد.

بعد از تغذیه مدل و اتمام یادگیری خروجی مانند ورودی به صورت دسته [۹] می باشد و باید به قولی Unbatch یا از دسته خارج شود و به حالت تصویر دربیاید. پس دوباره نیاز به پردازش هستند. سپس تصاویر را به همراه برچسب پیش بینی شده و برچسب آنها مقایسه می کنیم و همچنین می توانیم میزان پیش بینی برای هر نژاد را داشته باشیم. اینکار را با استفاده از رسم نمودار انجام می دهیم.

---

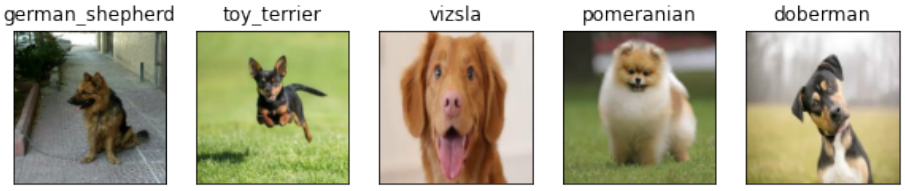
[۹] Batch



طبق تصویر مدل ما به درستی آموزش دیده و آماده است تا کل داده های آموزشی به آن تغذیه شود. بعد از اتمام آموزش مدل آموزش دیده را ذخیره می کنیم. ما از ابزار colab استفاده می کنیم و این ابزار یک سرویس رایانش ابری است که به صورت رایگان در دسترس می باشد ولی محدودیت هایی هم دارد. آموزش شبکه های عصبی و تکنیک یادگیری عمیق پردازش های سنگینی دارند و نیازمند کارت گرافیک قدرتمند هستند و اگر من از مدل را روی سیستم شخصی خودم اجرا می کردم مدت خیلی زیادی طول می کشید تا روند یادگیری تمام شود. بنابراین از سرویس رایانش ابری استفاده می کنیم. این سرویس محدودیت هایی دارد برای مثال سروری که در اختیار ما قرار می دهد بعد از مدتی که بلااستفاده می ماند از دسترس ما خارج می شود، یا حافظه و فضای ذخیره سازی آن محدود است. به

همین دلیل اگر اتصال ما از بین برود و دوباره به سرور آن متصل شویم باید تمام کدها را دوباره اجرا کنیم. با ذخیره مدل می توانیم دوباره آنرا بارگذاری کنیم و از آن استفاده کنیم.

پس از پایان این مرحله دیتای تستی را آماده می کنیم و با استفاده از مدل روی آنها پیش بینی انجام می دهیم. می توانیم نتیجه را در Dog Breed Identification from Kaggle Competition آپلود کنیم تا میزان دقت آنرا مشاهده کنیم. همچنین می توانیم تصاویر دلخواهی را به مدل تغذیه کنیم و مدل نژاد آنها را برای ما حدس یزند.نمونه تصاویری که توسط الگوریتم نژاد آنها پیش بینی شده است.



سپاس از زمانی که صرف کردید.

نوید کنعانی

منابع

MobileNet-V1, TowardsDataScience

MobileNet-V2, TowardsDataScience

Google Blog