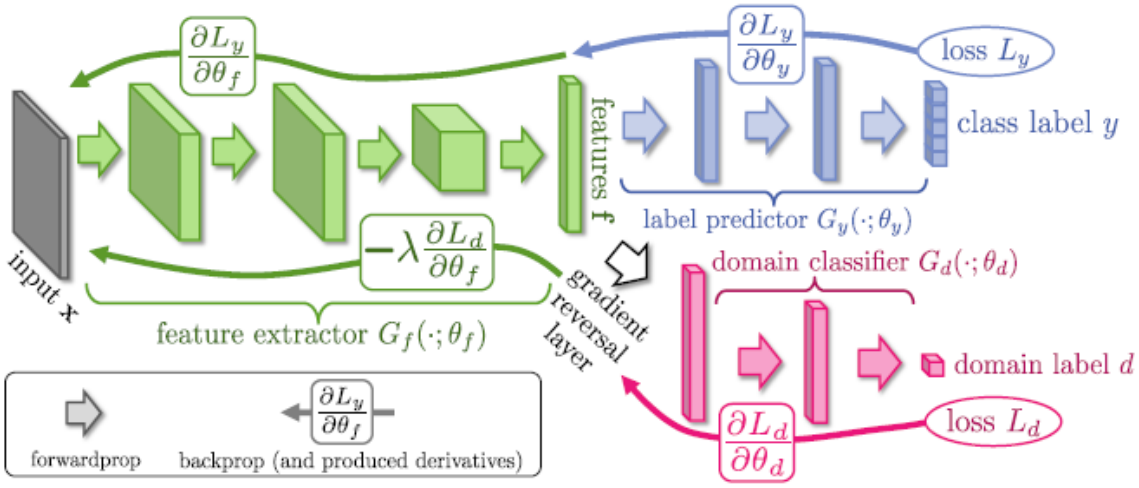
.۱



معماری پیشنهادی شامل یک استخراج کننده ویژگی عمیق (سبز) و یک پیش‌بینی کننده‌ی برچسب عمیق (آبی) است که در کنار هم یک معماری feed-forward استاندارد را تشکیل می‌دهند. انطباق دامنه بدون نظارت با اضافه کردن یک طبقه‌بند دامنه (قرمز) متصل به استخراج کننده ویژگی از طریق یک لایه برگشت گرادیان (gradient reversal layer) که در طول آموزش مبتنی بر backpropagation گرادیان را در یک ثابت منفی مشخص ضرب می‌کند انجام می‌شود. در غیر این صورت ، آموزش به روش استاندارد پیش می‌رود و خطای پیش‌بینی برچسب (برای نمونه‌های مبدأ) و خطای طبقه‌بندی دامنه (برای همه نمونه ها) را به حداقل می‌رساند. برگشت گرادیان اطمینان حاصل می‌کند که توزیع ویژگی‌ها بر روی دو دامنه مشابه هستند (تا آنجا که ممکن است برای طبقه‌بند دامنه غیر قابل تشخیص باشد) ، بنابراین ویژگی‌های مستقل از دامنه ایجاد می‌شود.

فرض می‌کنیم که مدل با نمونه‌های ورودی $x \in X$ کار می‌کند که X فضای ورودی و برچسب‌های مشخص (خروجی) y از فضای برچسب Y هستند. ما مسئله‌ی طبقه‌بندی را در جایی که Y یک مجموعه متناهی است فرض می‌کنیم ($Y = \{1,2,...,L\}$) ، با این حال این رویکرد عمومی است و می تواند هر فضای برچسب خروجی را که دیگر مدل های عمیق feed-forward از عهده آن برمی آیند ، کنترل کند. علاوه بر این فرض می‌کنیم که دو توزیع $S(x, y)$ و $T(x, y)$ روی X و Y وجود داشته باشد که از آنها به عنوان توزیع مبدأ و توزیع مقصد (یا دامنه مبدأ و دامنه مقصد) یاد می‌شود. هر دو توزیع پیچیده و ناشناخته فرض می‌شوند و بعلاوه مشابه هستند اما دامنه یکی shift یافته دامنه‌ی دیگری است.

هدف نهایی این است که بتوانیم با توجه به ورودی x ، برای توزیع مقصد برچسب‌های y را پیش بینی کنیم. در زمان آموزش ، به مجموعه بزرگی از نمونه‌های آموزش از هر دو دامنه‌ی مبدأ و مقصد با توجه به توزیع‌های

حاشیه‌ای $S(x)$ و $T(x)$ دسترسی داریم. متغیر باینری d_i (برچسب دامنه) نشان می دهد xi از توزیع مبدأ (اگر $x_i \sim S(x)$ آنگاه $d_i = 0$ است) یا از توزیع مقصد (اگر $x_i \sim T(x)$ آنگاه $d_i = 1$ است). برای نمونه هایی از توزیع مبدأ ($d_i = 0$) برچسب های مربوطه $y_i \in Y$ در زمان آموزش شناخته شده اند. برای نمونه‌هایی از توزیع مقصد ، برچسب ها را در زمان آموزش نمی‌دانیم و می‌خواهیم چنین برچسب هایی را در زمان آزمون پیش بینی کنیم.
اکنون یک معماری عمیق feed-forward تعریف می کنیم که برای هر ورودی x برچسب $y \in Y$ و دامنه آن $d \in \{0,1\}$ را پیش بینی می کند. ما چنین نگاشتی را به سه قسمت تجزیه می کنیم. فرض می‌کنیم که ورودی x ابتدا توسط نگاشت G_f (یک استخراج کننده ویژگی) به یک بردار ویژگی D بعدی $f \in R^D$ نگاشت شود.
نگاشت ویژگی ممکن است شامل چندین لایه feed-forward باشد . بردار پارامترهای تمام لایه ها را در این نگاشت به عنوان θ_f نشان می دهیم یعنی $f = G_f(x; \theta_f)$. سپس ، بردار ویژگی f با نگاشت G_y (پیش‌بینی کننده‌ی برچسب) به برچسب y نگاشت می شود و پارامترهای این نگاشت را با θ_y نشان می‌دهیم. سرانجام ، همان بردار ویژگی f توسط یک نگاشت G_d (طبقه بند دامنه) با پارامترهای θ_d به برچسب دامنه d نگاشت می‌شود.

در طول مرحله یادگیری ، هدف ما به حداقل رساندن خطای پیش بینی برچسب در مجموعه آموزش قسمت مبدأ است و بنابراین برای نمونه‌های دامنه مبدأ پارامترهای استخراج کننده ویژگی و پیش‌بینی کننده برچسب به منظور به حداقل رساندن از خطای تجربی ، بهینه می‌شوند. این تمایز پذیری ویژگی‌ها f و عملکرد کلی خوب پیش بینی ترکیبی از استخراج کننده ویژگی و پیش بینی برچسب در دامنه منبع را تضمین می کند. در عین حال می‌خواهیم ویژگی‌های f مستقل از دامنه باشد. یعنی توزیع‌های $S(f)=\{G_f(x; \theta_f) \mid x \sim S(x)\}$ و $T(f)=\{G_f(x; \theta_f) \mid x \sim T(x)\}$ مشابه باشند. این امر باعث می شود دقت پیش‌بینی برچسب در دامنه مقصد برابر دامنه مبدأ باشد. با این وجود که f دارای ابعاد بالا است و توزیع‌ها نیز با پیشرفت یادگیری دائماً در حال تغییر هستند ، اندازه‌گیری عدم شباهت توزیع $S(f)$ و $T(f)$ آسان نیست. یک روش برای تخمین عدم تشابه ، بررسی خطای طبقه‌بند دامنه G_d است ، به شرطی که پارامترهای θ_d طبقه بند دامنه آموزش داده شده باشند تا بین دو توزیع ویژگی به روش بهینه تفاوت قائل شوند.

این مشاهده منجر به ایده ما می شود. در زمان آموزش ، برای بدست آوردن ویژگی‌های مستقل از دامنه ، ما به دنبال پارامترهای θ_f نگاشت ویژگی هستیم که خطای طبقه بند دامنه را به حداکثر می‌رساند (با شبیه ساختن دو توزیع ویژگی به اندازه‌ای که ممکن است) ، در حالی که به طور همزمان به دنبال پارامترهای θ_d از طبقه بند دامنه هستیم که خطای طبقه بندی دامنه را به حداقل برسانند. علاوه بر این ، در تلاشیم خطای پیش‌بینی کننده‌ی برچسب را به حداقل برسانیم.

$$E(\theta_f, \theta_y, \theta_d) = \sum_{\substack{i=1..N \\ d_i=0}} L_y\left(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y), y_i\right) -$$
$$\lambda \sum_{i=1..N} L_d\left(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d), y_i\right) =$$
$$= \sum_{\substack{i=1..N \\ d_i=0}} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1..N} L_d^i(\theta_f, \theta_d) \qquad (1)$$

L_y خطای پیش بینی برچسب و L_d خطای طبقه بند دامنه است ، در حالی که L^i_y و L^i_d نشان دهنده توابع متناظر خطا برای نمونه‌ی iام از مجموعه‌ی آموزش هستند. بر اساس ایده خود ، ما به دنبال پارامترهای $\hat{\theta}_f$ و $\hat{\theta}_y$ و $\hat{\theta}_d$ هستیم که نقطه‌ی زینی تابع (۱) را برگرداند:

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d) \qquad (2)$$
$$\hat{\theta}_d = \arg\max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)\,. \qquad (3)$$

در نقطه زینی پارامترهای θ_d طبقه بند دامنه خطای طبقه بند دامنه را به حداقل می رسانند (چون با علامت منفی وارد (۱) می شود) در حالی که پارامترهای θ_y پیش بینی برچسب خطای پیش بینی برچسب را به حداقل می رسانند. پارامترهای نگاشت ویژگی θ_f خطای پیش بینی برچسب را به حداقل می رسانند در حالی که خطای طبقه بند دامنه را به ماکزیمم می‌کنند . پارامتر λ تعامل بین دو هدفی که ویژگی‌ها را هنگام یادگیری شکل می‌دهند کنترل می‌کند.

برای جستجوی نقطه‌ی زینی در (2)-(3) می‌توان از که حل‌کننده‌های standard stochastic gradient (SGD) استفاده کرد.

.۲

الف) عامل برای اینکه پاداش بیشتری را به دست بیاورد باید با توجه به اطلاعات و تجربیاتی که در گذشته به دست آورده actionها را اتخاب می‌کند یعنی Exploitation در حالی که برای به دست آوردن همین اطلاعات و تجربیاتی برای آینده عامل باید actionهای جدیدتر را انجام دهد یعنی Exploitation.

عاملی که فقط Exploit و یا فقط Explore میکند، نمیتواند موفق عمل کند زیرا مثلا عاملی که فقط Exploit میکند اولا هیچ دانش و تجربه‌ی اولیه‌ای ندارد که بر اساس آن Exploit کند ثانیا اگر این دانش اولیه را هم داشته باشد ممکن است با توجه به آن‌ها پاداش کمی دریافت کند در حالی که اگر Exploitation می‌کرد می‌توانست دانش بهتری به دست آورد و پاداش بیشتری دریافت کند. در ضمن عاملی هم که فقط Explore میکند شاید به صورت تصادفی بتواند با انجام دادن یک action پاداش دریافت کند اما قطعا برنامه‌ای برای سود بردن از actionهای جدیدی که پیدا کرده ندارد.

ب) پاداش امتیازی است که از طریق آن به عامل feedback داده می‌شود تا عامل بداند actionی که انجام داده است چقدر خوب یا بد بوده است. پاداش پس از action عامل به او داده می‌شود. اما ارزش یک وضعیت متوسط بازده تخفیف خورده است که عامل می‌تواند با شروع از آن وضعیت و سپس عمل کردن مطابق policy ما بدست آورد. در واقع ما ارزش یک وضعیت را مطابق یک policy به دست آورده و با توجه به آن انتخاب می‌کنیم یک action انجام داده و به آن وضعیت برویم یا نه.

پ) روشهای مبتنی بر Policy عامل مستقیما یاد می‌گیرد با توجه به وضعیتی که در آن قرار دارد چه actionی را انجام دهد اما در روشهای مبتنی بر Value به صورت مستقیم Policy را یاد گرفته نمی‌شود بلکه عامل یاد می‌گیرد اگر از وضعیت فعلی به هر کدام از وضعیت‌های بعدی برود این وضعیت‌ها چقدر ارزشمند هستند یعنی یاد می‌گیرد که کدام یک از وضعیت‌ها ارزش بیشتری دارند. بنابراین به صورت غیر مستقیم Policy را یاد می‌گیرد.

ت)

$$R(\tau) = r_{t+1} + \gamma\, r_{t+2} + \gamma^2\, r_{t+3} + \cdots$$

$$R(\tau) = \sum_{k=0}^{\infty} \gamma^k\, r_{t+k+1}$$

$\gamma = 0.5$

$R(\tau_0) = 2 + 0.5 \times 0 + 0.5^2 \times (-1) + 0.5^3 \times 2 + 0.5^4 \times 8 = 2.5$

$R(\tau_1) = 0 + 0.5 \times (-1) + 0.5^2 \times 2 + 0.5^3 \times 8 = 1$

$R(\tau_2) = (-1) + 0.5 \times 2 + 0.5^2 \times 8 = 2$

$R(\tau_3) = 2 + 0.5 \times 8 = 6$

$R(\tau_4) = 8$

$\gamma = 0.1$

$R(\tau_0) = 2 + 0.1 \times 0 + 0.1^2 \times (-1) + 0.1^3 \times 2 + 0.1^4 \times 8 = 1.9928$

$R(\tau_1) = 0 + 0.1^1 \times (-1) + 0.1^2 \times 2 + 0.1^3 \times 8 = -0.072$

$R(\tau_2) = (-1) + 0.1^1 \times 2 + 0.1^2 \times 8 = -0.72$

$R(\tau_3) = 2 + 0.1^1 \times 8 = 2.8$

$R(\tau_4) = 8$

.۳

اگر از Data Augmentation استفاده نشود:

اگر شبکه را با استفاده از وزنهای اولیه تصادفی آموزش بدهیم آنگاه دقت بر روی داده تست حداکثر ۵۰ درصد خواهد شد و دقت روی داده‌های آموزش ۱۰۰ درصد خواهد بود مدل overfit می‌شود.

اگر از مدل پیش‌آموخته بر روی مجموعه داده ImageNet به عنوان Feature Extractor استفاده کنیم آنگاه دقت بر روی داده تست حداکثر ۷۰ درصد خواهد شد و دقت روی داده‌های آموزش ۱۰۰ درصد خواهد بود مدل overfit می‌شود. همانطور که می‌بینید دقت نسبت به حالت قبل ۲۰ درصد بهتر شد.

اگر چندین لایه انتهایی شبکه را fine-tune کنیم آنگاه دقت بر روی داده تست حداکثر ۷۸ درصد خواهد شد و دقت روی داده‌های آموزش ۱۰۰ درصد خواهد بود مدل overfit می‌شود. همانطور که می‌بینید دقت نسبت به حالت قبل ۸ درصد بهتر شد. برای fine-tuning ما fine-tune کردن بلاک‌های ۱۱ به بعد، ۱۲ به بعد و ۱۳ به بعد را بررسی کردیم که fine-tune کردن بلاک‌های ۱۲ به بعد از همه موثرتر بود.

اگر از Data Augmentation استفاده شود:

اگر شبکه را با استفاده از وزنهای اولیه تصادفی آموزش بدهیم آنگاه دقت بر روی داده تست حداکثر ۵۶ درصد خواهد شد و دقت روی داده‌های آموزش ۸۴ درصد خواهد بود مدل overfit می‌شود. همان‌طور که می‌بینید Data Augmentation در بهبود دقت تست و کاهش overfit شدن تاثیر نسبتا خوبی داشته است.

اگر از مدل پیش‌آموخته بر روی مجموعه داده ImageNet به عنوان Feature Extractor استفاده کنیم آنگاه دقت بر روی داده تست حداکثر ۷۵ درصد خواهد شد و دقت روی داده‌های آموزش ۹۵ درصد خواهد بود مدل overfit می‌شود. همانطور که می‌بینید دقت نسبت به حالت قبل ۱۹ درصد بهتر شد. همان‌طور که می‌بینید Data Augmentation در بهبود دقت تست و کاهش overfit شدن تاثیر نسبتا خوبی داشته است.

اگر چندین لایه انتهایی شبکه را fine-tune کنیم آنگاه دقت بر روی داده تست حداکثر ۸۶ درصد خواهد شد و دقت روی داده‌های آموزش ۹۹ درصد خواهد بود مدل overfit می‌شود. همانطور که می‌بینید دقت نسبت به حالت قبل ۱۱ درصد بهتر شد. برای fine-tuning ما fine-tune کردن بلاک‌های ۱۱ به بعد، ۱۲ به بعد و ۱۳ به بعد را بررسی کردیم که fine-tune کردن بلاک‌های ۱۲ به بعد از همه موثرتر بود. همان‌طور که می‌بینید Data Augmentation در بهبود دقت تست و کاهش overfit شدن تاثیر نسبتا خوبی داشته است.

در خروجی شبکه نیز استفاده از pollingهای avg و max بررسی شد که در حالات محدودی منجر به کمی بهبود می‌شد اما در اغلب اوقات یا باعث کاهش دقت در تعداد ایپاک‌های مورد نظر می‌شد یا تاثیر چندانی نداشت.

Optimizerها و learning rateهای متنوع‌تری نیز تست شدند اما بهبود چندانی در دقت مشاهده نشد.

برای مشاهده عمکرد بعضی از مدل‌ها فرایند آموزش تا ۲۰ ایپاک هم کافی بود اما زمانی که داده‌ها augment شده بودند آموزش تا ایپاک‌های بیشتر الزامی بود.