今回の 2021 年眼科 AI コンテストで工夫した点についてです。

model は以下の２種類の転移学習モデルを使用しました。

１種類目の model は、
swin_base_patch4_window12_384_in22k
　(Swin-Transformer、ImageNet22k pre-trained)
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows
https://arxiv.org/pdf/2103.14030.pdf
の学習済みモデルを、PyTorch の timm を使用して転移学習しました。
Swin-Transformer は、Transformer をベースにして shifted windowing scheme という手法を使用した新しいモデルです。
CNN を使用せず、EfficientNet とほぼ同程度の精度が得られます。

２種類目の model は、
tf_efficientnetb7_ns
　（Efficient Net B7、NoisyStudent pre-trained）
https://arxiv.org/pdf/1911.04252
の学習済みモデルを、PyTorch の timm を使用して転移学習しました。kaggle でよく使用されている CNN のモデルです。

２種類のモデルの原理は全く異なりますが、得られた結果はほとんど同等の精度でした。その２種類のモデルの結果を複数回施行して mix して提出しました。その詳しい手順は後述致します。

小さい model の version でハイパーパラメータを探索して、大きいモデルで実際の学習をという作業を繰り返しました。

optimizer はいずれも Adabelief を使用しました。
AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients

https://arxiv.org/pdf/2010.07468.pdf
Adabelief は、Adam のような早い収束性・安定性と、SGD のような高い性能を兼ね備えた、新しい optimizer です。
複数の optimizer を様々な条件で試してみましたところ、今回の学習の場合には AdaBelief > AdamW > Adam > SGD という結果になり、また、SAM, AdaptiveSAM では良い結果が得られませんでした。
AdaBelief では eps=1e-16、weight_decouple=True, rectify=True という設定が最も良い結果が出ました。

最初から大きい learning rate にしますと、局所の最適解に陥ってしまい最終的な score が悪くなってしまう恐れがありますので、最初は小さい learning rate から開始し、徐々に増やし、その後漸減しました。validation のデータをモニターし、best score が出た時点で model を保存し、patience を超えた時点で自動的に best score のデータを読み出して小さい learning rate から漸増・漸減して再度学習させるようにしました。処理速度向上のため、model の保存・読み出しは RAMdisk に行いました。scheduler には CyclicLR を使用しました。

Batch size は４が最も良いデータが得られる事がある反面、ばらつきも大きいようでした。Batch size を 8 にすると安定する反面、良いデータが得られにくかったです。Batch size ４を複数回施行し、best なデータを取得する事で解決しました。

head（最終出力）部分は、その学習内容に依存しておりますので取り替えました。
単純にモデルからの出力を直接 1 にするよりも、
出力を 256 → 64 → 8 → 1 等のように Linear 結合の GELU[https://arxiv.org/pdf/1606.08415.pdf] を使用し漸減した方が良い結果が得られました。その理由を考えますと、画像の分類問題であれば出力を分類の数に直接固定するのが一般的ですが、今回は年齢を推定する問

題ですので、複数の出力からの様々な要素を複合したほうが年齢を推定しやすいという事なのではと考えました。
過学習を避けるため、間に drop も挟み、weight decay も使用しました。
また、そのまま普通に fine-tuning してしまうと転移学習の元の学習内容が破壊されてしまう可能性がありますので、まず本体の学習を全て freeze し、取り替えた本体の head の部分のみを学習させました。
その次に出力寄りの layer のみを順次 freeze を解除し、learning rate を下げながら学習させて行きましたが、head 本体の学習後に全部を同時に freeze 解除したデータと比較しても改善は得られませんでした。その理由としては、pre-trained での学習内容と、今回の眼底の学習内容が大きく異る為ではないかと考えました。

LayerNormalization の layer は fine-tune によって元の学習内容が特に破壊されやすいので、layer の学習を freeze しました。
TensorFlow と違い、PyTorch ではそれを自分で実装する必要がありました。

学習速度を上げるために、Mixed Precision を使用しました。16bit の計算で済みますので、計算が高速になり、GPU のメモリ使用量も節約できます。計算結果にはほとんど影響はありません。

Dataset, Data Loader は自分で実装しました。
epoch 毎にハードディスクから画像データを読み込んでリサイズをすると非常に時間がかかるため、予め画像データを中間サイズの画像にリサイズしてメモリ上に全て置いておき、そこから読み込むようにしました。

画像の処理についてです。

まず予め周囲の黒い部分を取り除き、正方形にしました。
その画像をそのまま使用するのではなく、学習させる画素数よりも大きめの画像から crop したほうが良い結果が得られ、色々試してみましたところ学習させる画素数の 1.15 倍(efficientNet B7)～1.2 倍(Swin)の画像からの crop が最も良い結果が得られました。（その倍率が高いほど、４隅の黒い部分が少なくなって高解像度の画像を model へ送る事ができる反面、上下左右の周辺側の画像が切り捨てられる事になります。もともとの処理できる画素数が高い model（efficientnNet B7＞ Swin)ほど、周辺の切り捨てられる情報を少なくしたほうが良いデータが得られるという事と思います。）

Data Loader にはメモリ上のその画像を Random Crop して渡し、validation / test 時には Center Crop しました。そのため、単一の施行では周辺の画像が切り捨てられる部分が発生しますが、複数 epoch 行う事によって学習時には画像の全ての部分を学習する事になります。逆に、tran / val 時には周辺側の画像が固定して切り捨てられます。
その他にも学習時には Random Horizontal Flip / Random Rotate / Random Zoom も行いました。test 時には Horizontan Filp も併用しました。transform の処理は CPU よりも全て GPU で行った方が高速でした。

今回の提供して頂いた画像を見ますと、同じ患者さんの同一眼のデータが複数ある事に気が付きました。
そのまま train / validation にランダムに振り分けてしまいますと、同じ患者さんの画像が train と validation に同時に存在して不具合が生じてしまう恐れがありますので、validation に採用した患者さんの同一眼は train に採用しないようにしました。
本来は train / validation / test と分けるのが理想ですが、今回はサンプル数の関係で validation と test は兼用としました。

mean squared error(MSE)を小さくするために、validation 及び最

後の提出用のデータの最後の age の出力の段階で、

・普通に age の出力
・左右を Horizontal Flip してから age の出力

を行い、その平均値を age の出力としました。

その後、同一の ID の患者さんのデータを左右眼とも pickup し、
その年齢を平均したものを出力しました。

さらに、上記の学習プロセスを複数モデルでそれぞれ複数回繰り返し、その重み付け平均（加重平均）を作成しました。加重平均の重み付けは MSE の 30 乗の逆数が best でした。

その際、train / validation をその学習プロセスの度にランダムに選択してしまうと、最適な加重平均の算出に影響が出てしまいますので、train / validation は固定して共通して使用するようにしました。

但し、Train / validation を固定してしまいますと、validation の選択による最終結果のばらつきが生じますので、複数回の学習後、
train / validation を再度シャッフルし直して固定してから学習を再度複数回施行し、得られた結果の加重平均を作成し、それらの最終的な加重平均を提出用のデータとしました。

計算の流れとしては、
モデルからの出力→Horizontal Flip の平均→同じ ID の平均→複数モデルの加重平均→train / val を再設定後の加重平均という形です。

以下は他にも試行した内容です。

転移学習を高速に行うため、画像データを転移学習 model に入れてシリアル化して head の出力を保存し、head 以降の部分だけのモデルを作成してそのシリアル化したデータを学習させるという方法も試してみました。転移学習の fine-tune off で学習させた場合に施行する内容と同じ事を、１回のみ施行してそれをファイルに保存して再利用するというアイデアです。この方法ですと、転移学習元の model での計算は最初の１回だけで済みますので、圧倒的に低コスト（時間的には数十～数百分の１程度）で学習できます。この方法を利用して、head 以降の部分の最適な部分 model の構築を調べるために、部分 model の学習を数百 epoch x 数百回繰り返し、その部分 model の平均値を算出する事で、最適な部分 model の推測ができました。それだけでは学習効果は今一つですので、その後、本体の model を含めた転移学習を行いました。実際には本体の転移学習を行いますと、それに応じて head 以降の最適な model も異なって来ますので、あくまでこの方法は目安として利用しました。

また、学習する画像を予め CLAHE(Contrast Limited Adaptive Histogram Equalization)処理して、コントラストを上げた画像で学習を試行しましたところ、むしろ学習精度が下がってしまいました。写真のコントラストも年齢を推測する要因の一つなのかもしれません。

他にも、以下の転移学習 model を単独で使用したり、
並列結合して同時に学習したり、
単独学習後に MSE の逆数の平均を取ったり、色々試しました。

以下は試行して採用されなかった数々の model です。

・ tf_efficientnetv2_l_in21k、tf_efficientnetv2_m_in21k

（EfficientNetV2 L 及び M、 ImageNet21k pre-trained）

本来は EfficientNet B7 よりも良いデータが得られるという事のようですが、実際試してみて色々なハイパーパラメータでいじってみましたが、良いデータが得られませんでした。また、PyTorch では無く TensorFlow での original の EfficientnetV2 も try しましたが、mixed precision がうまく動作せずに遅く、また当然ながら PyTorch と同等のデータでした。

・ vit_small_r26_s32_384、vit_base_r50_s16_384

(Vision Transformer と Resnet の混合モデル、AugReg21k pre-trained。）

How to train your ViT? (https://arxiv.org/pdf/2106.10270.pdf)を見ますと、この混合モデルが眼底の分類タスクで最も良いデータが得られているとの記載でしたが、実際に今回のタスクで色々な条件で試してみましたところ、残念ながら良いデータが得られませんでした。

・ vit_large_patch16_384

(Vision Transformer Large)

同上の論文を見ると、classification の分野でとても良い結果が出ているようですが、今回のタスクでは良い結果が得られませんでした。

・ cait_s24_384

(CaiT: Going deeper with Image 、 ImageNet pre-trained)

処理が重く、また良い結果が得られませんでした。

・swin_large_patch4_window12_384_in22k

(Swin-Transformer、 ImageNet22k pre-trained）

はデータの数に比べモデルが大きすぎる為か、過学習を起こしてしまい良いデータが得られませんでした。 swin_base の方がなぜか良いデータが得られました。

・tf_efficientnet_l2_ns

(EfficientNet L2 noisy-student pre-trained)

最も良い結果が得られる事を期待したのですが、モデルが巨大な為、Mixed Precision の Batch2 でも GPU のメモリに乗り切らずにエラーが出て動作しませんでした。

・その他 ResMLP、gMixer、nfnet、deit 等、様々な転移学習モデ

ルも try しましたが、いずれも良い結果が得られませんでした。

また、以下の混合モデルも作成しましたが、良い結果が得られませんでした。

・２種混合モデル
(swin_base_patch4_window12_384_in22k と
tf_efficientnetv2_s_in21k の mix model)

・３種混合モデル
(swin_base_patch4_window12_384_in22k と
tf_efficientnetv2_s_in21k と
cait_s24_384 の mix model)

・４種混合モデル
(swin_base_patch4_window12_384_in22k と
tf_efficientnetv2_s_in21k と
cait_s24_384 と
vit_small_r26_s32_384 の mix model)

上記の結合する header 以降の部分を調整して試行錯誤してみましたが、いずれも単独のモデルの値を超える事ができず、むしろそれぞれ単独でのデータの出力の加重平均を取った方が良い数値が得られました。
その理由としては、単独のモデルであれば、EarlyStopping を使用して良い結果が得られたデータをリストアして再学習すれば良いのですが、混合モデルの場合には、同じ事を行うと混合モデル全体をリストアする事になってしまう為に、良いデータが得られにくいのかもしれません。

また、ハイパーパラメータの探索には最初はRayTuneも使用しましたが、多くのモデルはその決定時間がとても長くかかり、modelの型の大きな変更等はRayTuneでは対応できず、結局ほとんど手作業で行う事になりました。

感想：
もともと私は中学生の頃からコンピュータを使用していました。当時はパソコンではなくマイコンと呼ばれていて、愛読書は「マイコンベーシックマガジン（ベーマガ）」でした。お年玉で買った MSX で、MSX-basic でプログラミングを楽しんでいまして、家族や友達から変人扱いされていました。かれこれ 35 年ほど前の話です。
その後もパソコンやプログラミングには興味があり、医院のパソコンは全て自作、医院の電子カルテ及びシステム一式は C#(Visual Studio)及び python(PyCharm)で自作しておりますが、今まで Deep learning のコードを自分で作成した事はありませんでした。

今回、眼科 AI コンテストを開催して頂いた事をきっかけに、３ヶ月ほど前から実際に自分で Deep learning のコードをいじってみると非常に楽しく、時間を忘れて没頭している毎日です。
自院の Supermicro M/B Xeon マシン(自作) の VMware ESXi(無償版)のサーバーに NVIDIA RTX3090 を追加し、メモリも追加し、ubuntu20.04 の仮想マシンに GPU パススルーして使用しました。
自宅からも医院のサーバーをいじりたいですので、ESXi 上の ubuntu 仮想マシンに OpenVPN を楕円曲線暗号の設定（最も安全性が高いです）でインストールし、NTT 東日本のフレッツ網内折返し IPv6 を使用した OpenVPN トンネルの中で VMRC を使用して快適に接続しております。もちろん、遠隔操作時にも全てのデータは院内のみの安全な場所で完結しております。不要なポートは全て塞ぎ、不要なサービスは起動せず、セキュリティ対策には特に注意しております。

最初は TensorFlow から始め、自分で１から学習させるよりも転移学習の方が良い結果が出る事に気づき、もっと色々と自分でカスタ

マイズしたいと考えて PyTorch に移行し、timm を使用し様々なモデルを転移学習して try してみました。TensorFlow よりも PyTorch の方が自分で色々いじれるので使用していて楽しいです。IDE は PyCharm を使用しております。

まだ機械学習に関しては今回のコンテスト向けの勉強しかしておりませんので、今後はさらに他の deep learning の学習を続けて、kaggle への参加や、数学の勉強、OCT の画像等からのオリジナルの学習等を試してみたいと思います。開業医ですので、精度の高い学習データが少ないのが一番の問題ですが・・・。

今回の AI コンテストは、私にとって deep learning の勉強がとても楽しいという事に気づかせてくれました。このような学習の機会を与えて頂きました、日本眼科 AI 眼科学会及び関係者の皆様に深く感謝を致します。有難うございました。
もし可能でしたら、来年もぜひ何らかの別の AI コンテストを開催して頂ける事を楽しみにしております。

2021 年 10 月

〒299-1162
千葉県君津市南子安 2-8-30
南子安眼科
古山　誠
e-mail: minamikoyasuganka@gmail.com
tel : 0439-27-1022