



Animal Imbalance Classification Round3

~in class コンペにおける解法の説明~

LB:0.8352313147863246

Private:0.8411414625726732

自己紹介



名前: 坂梨 (Kaggle名: Stefano) from team 1

職業: 機械系エンジニア→データアナリスト(コンサルティング業界)

→データサイエンティスト(現在、某通信キャリア常駐でデータアナリスト寄りの業務)

Kaggle実績

- PetFinder - Pawpularity Contestに参加(画像)

→ブロンズ圏内(チームメンバのプライベートシェアリングによりメダル剥奪)

- Feedback Prize - Evaluating Student Writing(NLP) 参加中

- HappyWhale - Whale and Dolphin Identification(画像) 参加中

趣味: 海外旅行、Kaggle、息抜きにゲーム

Kaggleリンク: <https://www.kaggle.com/stefanojp>

Agenda



- 本コンペにおける制約
- 制約への取り組みかた
- 技術的な部分についての解法
- やり残したこと、やりたかったこと

3つの制約



- 時間的制約

平日は働いている、他コンペもやっているのでコンペに当てれる時間が限られていた

平日は夜2～3時間、土日は半日ぐらい

- 人的リソースの制約


チームメンバーは自分含めて4人

メンバーで目的も環境も異なるので、チームでどのように取り組むか

- GPUリソースの制約

Kaggle kernel上のGPUリソースをリアルコンペ用に確保しときたい

制約に対する取り組み方

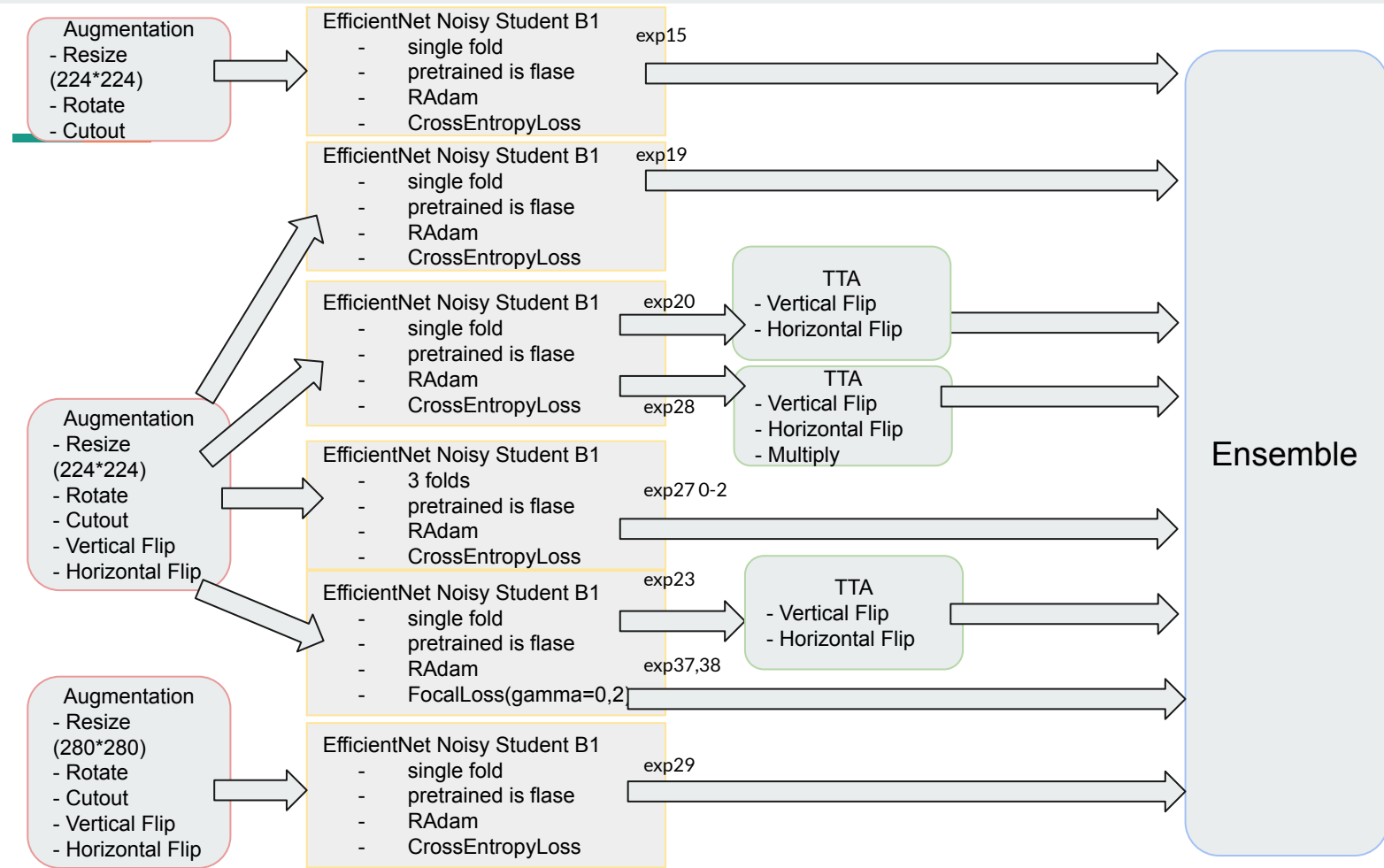
- 
- 時間的制約に対して・・・
 - ・Week1のDigit Recognizerコンペで、今回のベースラインとなるコードの理解と作成
 - ・in classコンペが開始したら、寝る前にコード回しとして、朝起きたら提出できるようにした
 - 人的リソースの制約に対して・・・
 - ・Slackにてチームメンバー招待
 - ・わからないこと質問しあう
 - ・コードシェア
 - GPUリソースの制約に対して・・・
 - ・Google Colab Proを契約しているので、Colab上でコードを実行できるようコード改修
 - ・同時に2つのコードを実行できる、(+Kaggleで回せるので3つ)

技術的な取り組み

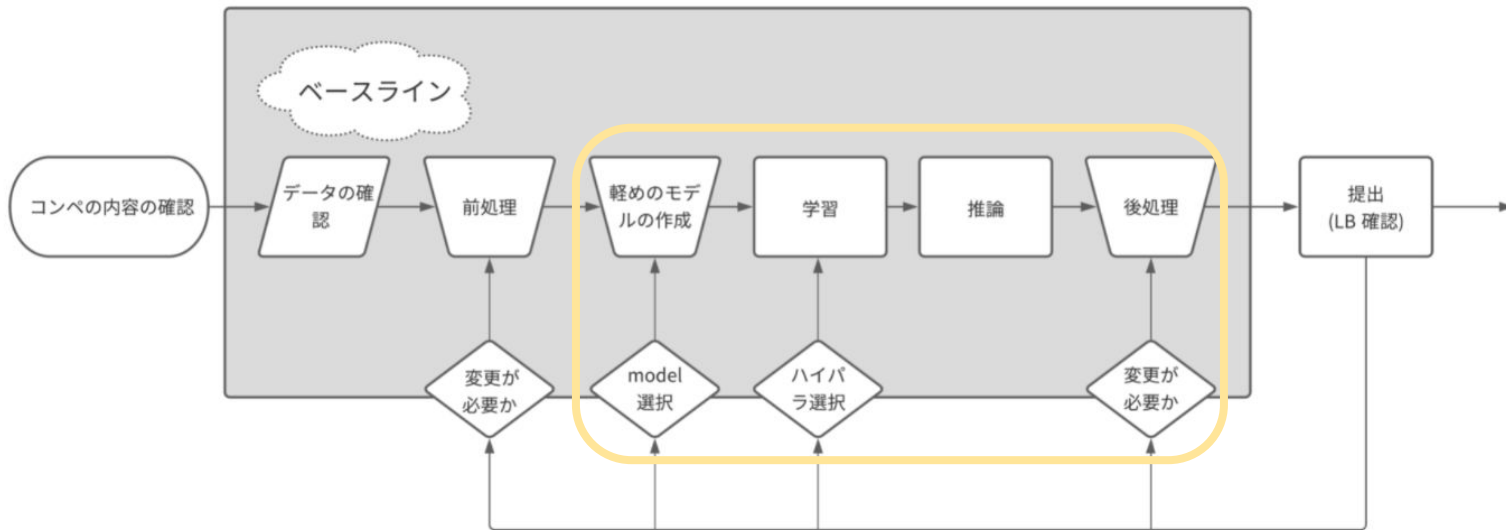
- 過去画像系PetFinderコンペの経験から、複数のアーキテクチャを使用して特徴量を抽出したモデルのアンサンブルが強いという仮説からスタート
- ResNetのみならず、EfficientNet、EfficientNet Noisy Student、Swin-Transformer、ViT、NfNetなどを試した
- そのほかにも、DataAugmentationにおけるMixupやinference時のTTA、Optimizerの検討、Fold、Loss関数の検討を試した

| A | B | C | D | E | F |
|-----|--------------------------------------|--|---------|-----|---------|
| No. | 実装内容 | 詳細 | 予定日 | 完了日 | 対応者 |
| 1 | Stratified KFold | データ量が小さくなるため、精度があがらない | 3/1 | 3/1 | Stefano |
| 2 | Colab用インプットデータリンク先修正 | Colab訓練時タイムアウトエラーを引き起こしている | 3/2 | 3/1 | Stefano |
| 3 | TTA | 一旦実装完了、結果を見て確認する | 3/6 | | |
| 4 | アンサンブル用推論のみコード | 同じアーキテクチャならアンサンブルできる仕様 | 3/2 | 3/2 | |
| 5 | Early Stopping | エポックを増やせば精度上がる傾向があるため | | | |
| 6 | MixUp | モデルによっては大きな向上につながる→高い精度につながらなかった。訓練時間ものびた | | | |
| 7 | 画像からの特徴量抽出 | YOLOによる物体検出、品種など | | | |
| 8 | SVR Head?? from petfinder | | | | |
| 9 | Swin-V2モデル | bsを大きくできず、精度があがらない。モデルが大きすぎる。 | 3/2 | 3/2 | stefano |
| 10 | exp6とexp8の違い | Colab vs Kernel,実行環境の違いのみ、アンサンブル時にexp8モデル使う | 3/2 | | Stefano |
| 11 | resnetとイメージサイズの関係 | | | | |
| 12 | optimizerの変更AdamW | AdamからAdamW,結果まち | 3/2 | 3/3 | Stefano |
| 13 | optimizerの変更RAdam | AdamからRAdam、resnetにおいてはRAdamが精度良い | 3/2 | 3/3 | Stefano |
| 14 | self.cls = nn.Sequential | | | | |
| 15 | EfficientNetの実装 | E0,E0_nsのみbs=128で実行可能。E1_nsはbs=64で実行可能 | 3/2&3/3 | 3/2 | Stefano |
| 16 | EfficientNetモデルとのアンサンブル | ResNetモデルとEfficientモデルのアンサンブルでエラー、efficientnetモデルおそらく、basemodelがresnetのままのため | 3/5&3/6 | | |
| 17 | EfficientでのOptimizer比較 (AdamWとRAdam) | RAdamの方が良い精度が出やすい | 3/3&3/4 | 3/4 | Stefano |
| 18 | YOLOによる物体検出モデル | ラベルづけ、ここで訓練済みモデル使うのは禁止？ | | | |
| 19 | Stratified KFoldの見直し | 現在のFoldの分け方であれば、train,testデータセットでのラベルの比率は同じ | | 3/6 | Stefano |
| 20 | 損失関数の見直し、Focalloss | 効果は薄い,crossentropylossで問題なし | | 3/6 | Stefano |
| 21 | inference code にTTA | | | | |

ベストモデルのアーキテクチャ(0.8411414625726732)

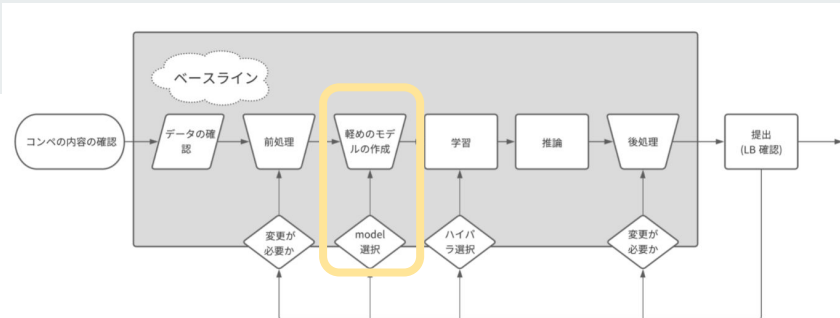


主に取り組んだパート



(参照: かぐら座Week2 Notebookから)

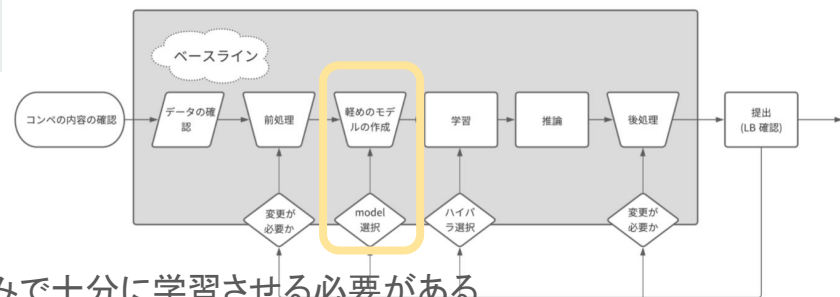
試したアーキテクチャ



- ResNet34 vs ResNet50
- EfficientNet vs EfficientNet Noisy Student
- Swin-Transformer, Vision Transformer
- NfNet

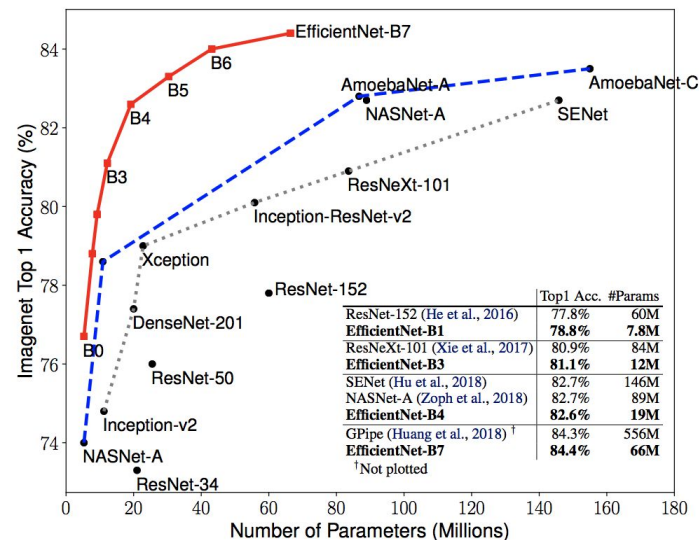
| | A | B | E | F | G |
|----|--------|---|--------------|---------|-----------------------|
| 1 | exp | detail | public | private | model |
| 4 | exp1-1 | resnet50, decrease image size to 64 | 0.4924956869 | | ResNet50 |
| 5 | exp2 | add vertical flip and horizontal flip based on exp1 | 0.4571976303 | | ResNet34 |
| 6 | exp3 | KFold baseline (test submission) | | | |
| 7 | exp4 | fix import file path (test submission) on no fold version | | | ResNet50 |
| 8 | exp5 | fold | 0.4514109725 | | |
| 9 | exp6 | no fold version - large batch size 128 and change min lr | 0.7854885555 | | ResNet34 |
| 10 | exp7 | large batch size with 3 k folds | 0.3898602868 | | ResNet34 |
| 11 | exp8 | no fold version - large batch size 128 for Colab, exp6をcolabで実行 | 0.7967354787 | | ResNet34 |
| 12 | exp9 | same as exp6 for inference use | 0.7854885555 | | ResNet34 |
| 13 | exp10 | based on exp8 change to AdamW | 0.7913599719 | | ResNet34 |
| 14 | exp11 | based on exp8 change to RAdam、RAdam採用する (Resnetだけかも) | 0.7963992563 | | ResNet34 |
| 15 | exp12 | ensemble exp8 and exp9 | 0.7965362484 | | |
| 16 | exp13 | ensemble exp8, exp9, exp10, exp11 | 0.8010762291 | | |
| 17 | - | Swin-T large 224, bs=128はOOM bs=32ぐらいでやっと動く | | | |
| 18 | - | Swin-T small 224,bs=128はOOM, 0.6ぐらいまでであがりきらない、bs | | | |
| 19 | exp14 | EfficientNetB0,publicが低い確認する | 0.7622874265 | | efficientnet_b0 |
| 20 | exp15 | EfficientNet_b1_ns,RAdam | 0.8005274522 | | tf_efficientnet_b1_ns |
| 21 | exp16 | EfficientNet_b0_ns,RAdam | 0.7564432525 | | tf_efficientnet_b0_ns |

試したアーキテクチャの結果 ResNet系



- 今回Pretrained がFalseであり、用意されたデータのみで十分に学習させる必要がある
- より多くのバッチサイズ、より大きな画像サイズを使いたい
- バッチサイズが大きすぎるとOOM(128以上)
- バッチサイズを小さくして、モデルサイズが大きいモデルを採用するとOOMもしくは精度あがらない
- 画像サイズを下げてバッチ数をあげようとしたがOOMすることも多かった
- よって、大きなアーキテクチャは使えないことがわかった

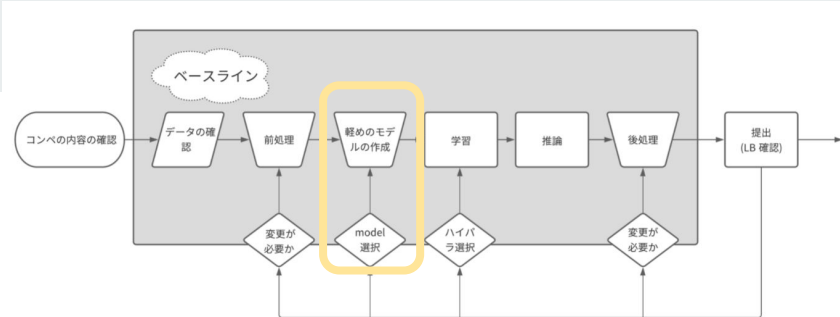
- 例えば、ResNet50ではイメージサイズ64×64、バッチサイズは64
- 一方でResNet34ではイメージサイズ224×224、バッチサイズは128で回すことができた
- この時点で精度は0.7後半



(参照)EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

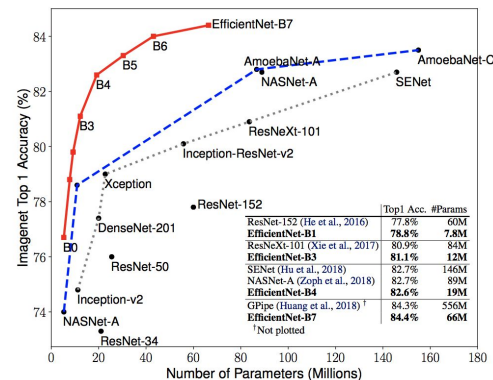
試したアーキテクチャの結果

EfficientNet系



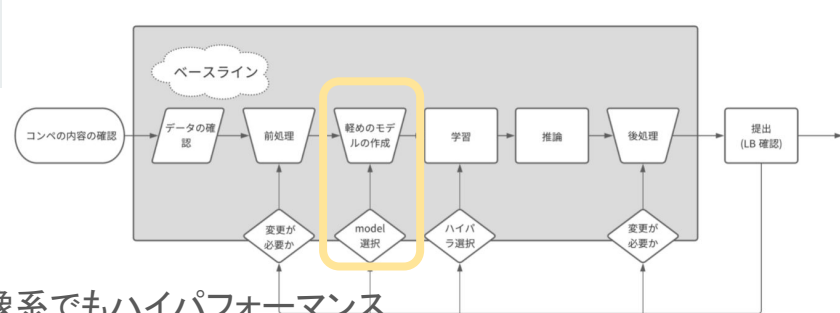
- EfficientNetB0からB7までトライ
- 実行できかつ精度がある程度担保されたのはB1まで
- ResNet同様、モデルサイズ小さいB0を使用して、
イメージサイズとバッチサイズを大きくした方が精度がよかった
- 精度0.7後半ぐらい

- 過去コンペで試して精度の良かったEfficientNetNoisyStudentをトライ
- 画像サイズ384モデルではOOM
- 画像サイズ224バージョンでNS_B0とNS_B1をトライ
- NS_B1でもB0と同バッチサイズで実行でき精度向上がみられた
- 精度0.8ぐらい



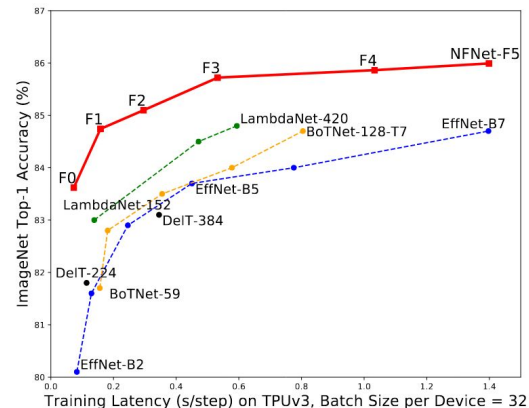
(参照)EfficientNet: Rethinking Model Scaling
for Convolutional Neural Networks

試したアーキテクチャの結果 Transformer系 & NfNet



- NLPで高い精度を出すのに貢献したTransformer、画像系でもハイパフォーマンス
- PetFinderコンペで多くの方が実装したSwin-Transformerをトライ
- 同じくtransformer系のVision Transformer (ViT) もトライ
- どちらも事前学習にかなり大きなデータ量が必要であることが知られている
- そのため、上げれる最大バッチサイズで訓練するも、一向に精度があがらない(0.6以下)
- 学習率を調整しても訓練時間がほかモデルに比べ長い(ため断念1epoch、30min)

- 今回初めて使用するNfNet
- 訓練時間は1epochに25分掛かるが、最初から精度が高い傾向
- アンサンブル用に実行
- 結果アンサンブル時に使用できなかった



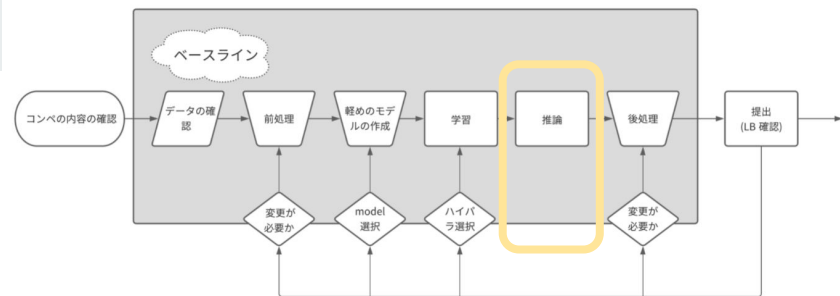
ハイパーパラメーターの調整



- 最終的なパラメーター一覧は下表(今回はあまりハイパラ調整を重視しなかった)
- PetFinderコンペで効果のあったMixup
- Mixupは α を調整するも訓練時間が増加し、精度の向上につながらなかった
- モデルによってはFlip系を入れると精度が下がった

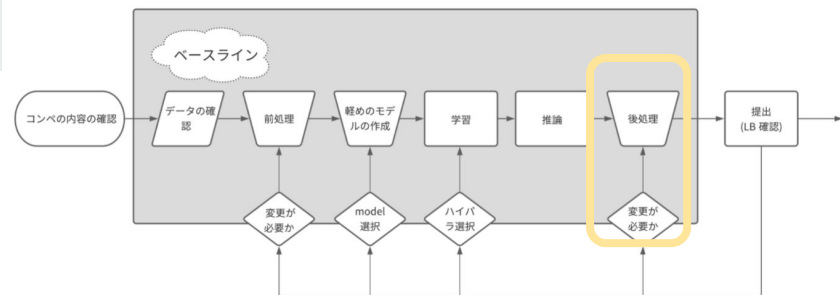
| 試したこと | detail |
|--------------|---|
| lr | 1e-5 |
| min lr | 1e-3 発散しない程度に幅をもたす |
| epoch | 50(モデルによっては20) 訓練時間が9時間以内、Foldとの兼ね合いも |
| image size | 224(280) |
| batch size | 64(128は基本OOM) |
| loss | CrossEntropyLoss(一部、FocalLoss) |
| Augmentation | Resize, Rotate, cutout, (Horizontal & Vertical Flip), (Mixupは不採用) |
| Optimaizer | Adam→AdamW→RAdamWを採用 |

推論での工夫



- こちらもPetFinderで効果のあったTTAを実装
- Pytorchでの実装が難しく、API(ttach)を利用
- <https://github.com/qubvel/ttach>
- 精度には効いてそうな、効いてなさそうな(検証方法が確立していない)
- (もしかしたら実装にバグがあるかも?)

後処理

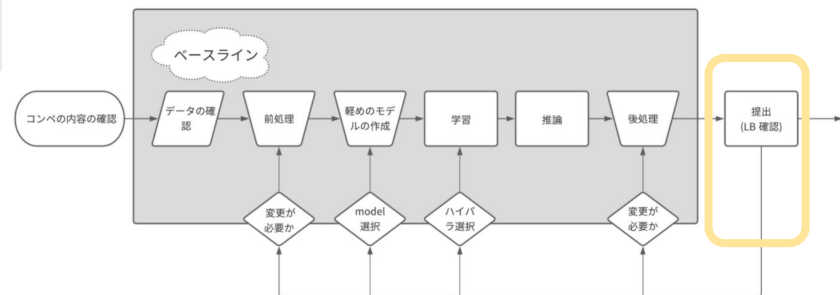


- CVとPrivateでのAccuracyに差がある
- 大きい時は0.09程度
- StratifiedKFoldによる汎化性能をあげる取り組み
- 訓練時間を考慮して3Foldによるアンサンブル、CVとPrivateによる差を縮めることはできた
- しかしながら、精度向上に大きく繋がるようなことはなかった

- 複数モデルでのアンサンブル
- 各モデルごとに重みはつけていない
- (同じアーキテクチャモデル同士でのアンサンブルしかできていない)
- 3モデルのアンサンブルによって精度向上がみられた
- よって最終11モデルでのアンサンブル

| Test Dataset | Private |
|--------------|--------------|
| 0.8764383562 | 0.8005274522 |
| 0.8537 | 0.7564432525 |
| 0.8689041096 | 0.7809556327 |
| 0.8765753425 | 0.7922304446 |
| 0.8939726027 | 0.8335906151 |
| 0.8915068493 | 0.8180110004 |
| 0.7850684932 | 0.6506512947 |

結果

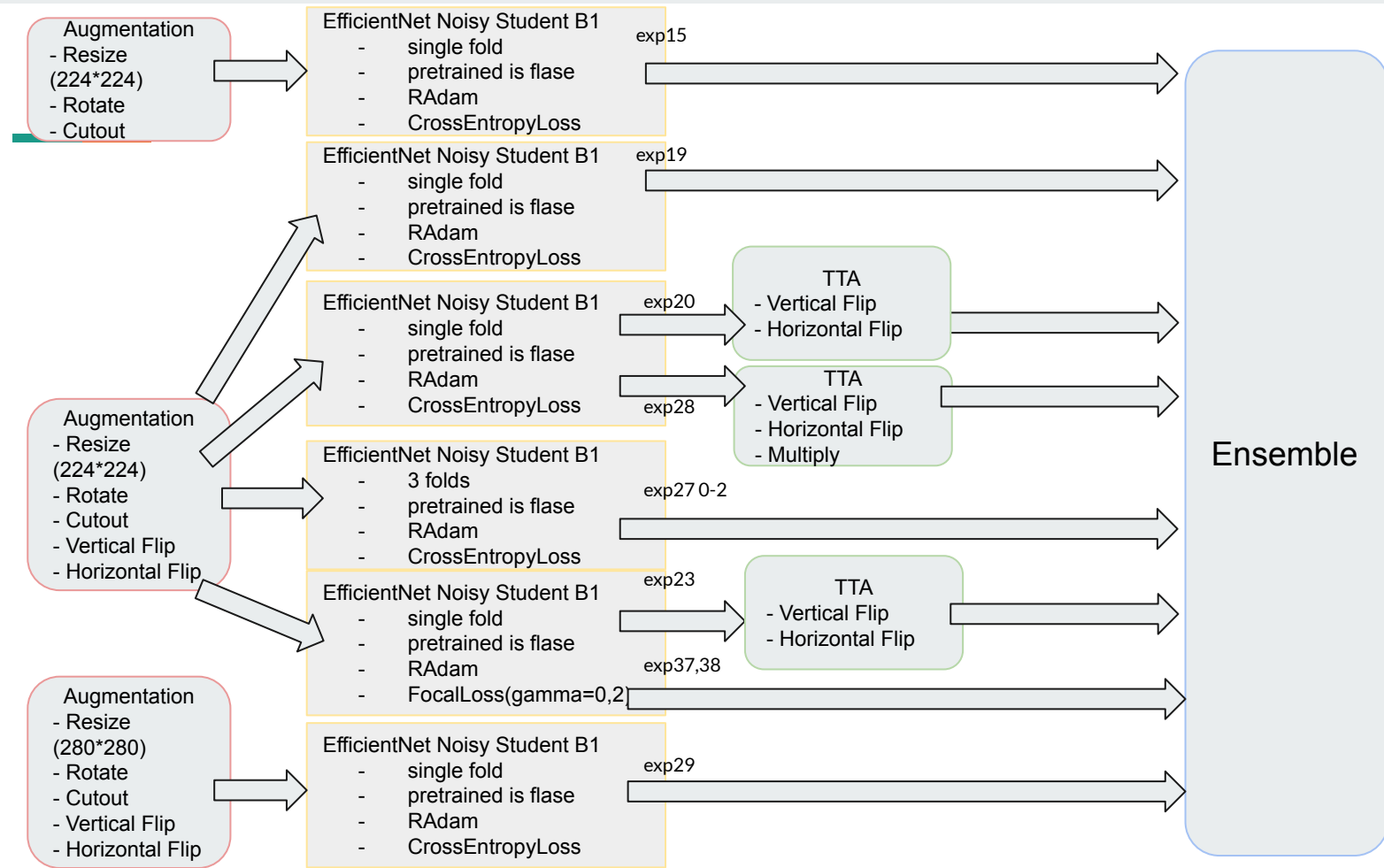


- 計40のアウトプットcsvを提出
- 1つのアーキテクチャでのシングルフォールドモデルよりアンサンブルによって精度向上した
- 11モデルのアンサンブル (EfficientNet Noisy Student B1ベース) による Accuracyが0.8352313147863246で終了


| Date | Score |
|----------|--------------------|
| 2022/3/1 | 0.5090491575336402 |
| 2022/3/2 | 0.7967354786896385 |
| 2022/3/3 | 0.8010762290836059 |
| 2022/3/4 | 0.8010762290836059 |
| 2022/3/5 | 0.8180110003671409 |

| Date | Score |
|-----------|--------------------|
| 2022/3/6 | 0.833590615075277 |
| 2022/3/7 | 0.833590615075277 |
| 2022/3/8 | 0.8349676012722952 |
| 2022/3/12 | 0.8352313147863246 |

ベストモデルのアーキテクチャ(0.8411414625726732)



やり残したこと

- 
- 別のアーキテクチャモデルで作成したモデルでのアンサンブル
EfficientNet Noisy StudentとResNetとNefNet
 - アンサンブル時の重みつけ
 - 正しいTTA?
 - YoloV5による物体検出から得られたラベルを特徴量として使用するモデル
 - Data Augmentationの調整
 - ハイパラの調整