初Kaggle & 初メダル獲得🥉
Google - Isolated Sign Language Recognitionに参加して

2023/5

Agenda

- Kaggleとは？
- コンペ概要
- 取り組み内容

Kaggleとは？

■ **Kaggle**

- 企業や研究者がデータを投稿し, 世界中の統計家やデータ分析家がその最適モデルを競い合う,
  予測モデリング及び分析手法関連プラットフォーム.（Googleの子会社）

- 議論やコードは可能な限り積極的にシェアされ, よりよい方法を学ぶことができる.

- 基本的にお題とデータが与えられ, それに沿ったモデルを提出し, その予測精度を競う.
  上位に入賞すると, 実際にモデルが実課題へ適用されたり, 賞金がもらえたりする.

- 世界ではNVIDIAやH2O.AI, 日本では,PFNやDeNAのエンジニア, 研究者が多い.
  企業によってはKaggleのランキングで採用したり, 業務時間のXX%をKaggleに充ててよいなどルールあり.

■ **Kaggleと私**

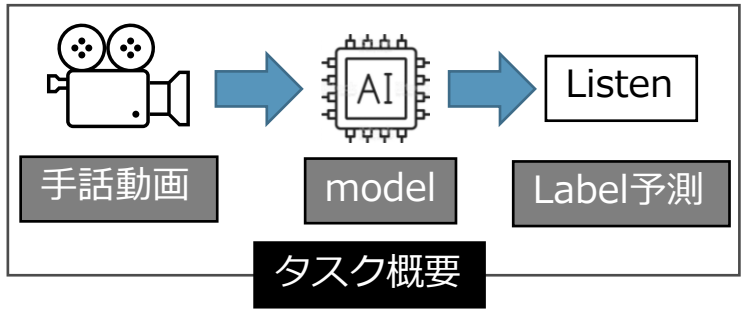
- 有名なTitanic生存者予測は経験済み.（Kaggleやる人の登竜門的なチュートリアル）
  辞書的に”Kaggleで勝つデータ分析の技術”を業務で使用.
  （業務に通用するノウハウがある気がする..特徴量の作り方, バリデーション方法等）

- 実際のコンペに参加するのは初めて.

Kaggle - Wikipedia
Titanic - Machine Learning from Disaster | Kaggle

コンペ概要

- **目標**
  isolated American Sign Language (ASL) signsを正しく分類すること.



- **背景**
  視覚障害を持つ赤ちゃんを授かった親は手話を覚えることが難しい.
  手話習得ゲームとしてAndroidに組み込み, 習得の効率化を目指す.
  ゲーム画面で与えられた単語に該当する手話をユーザがしているか判定したい.

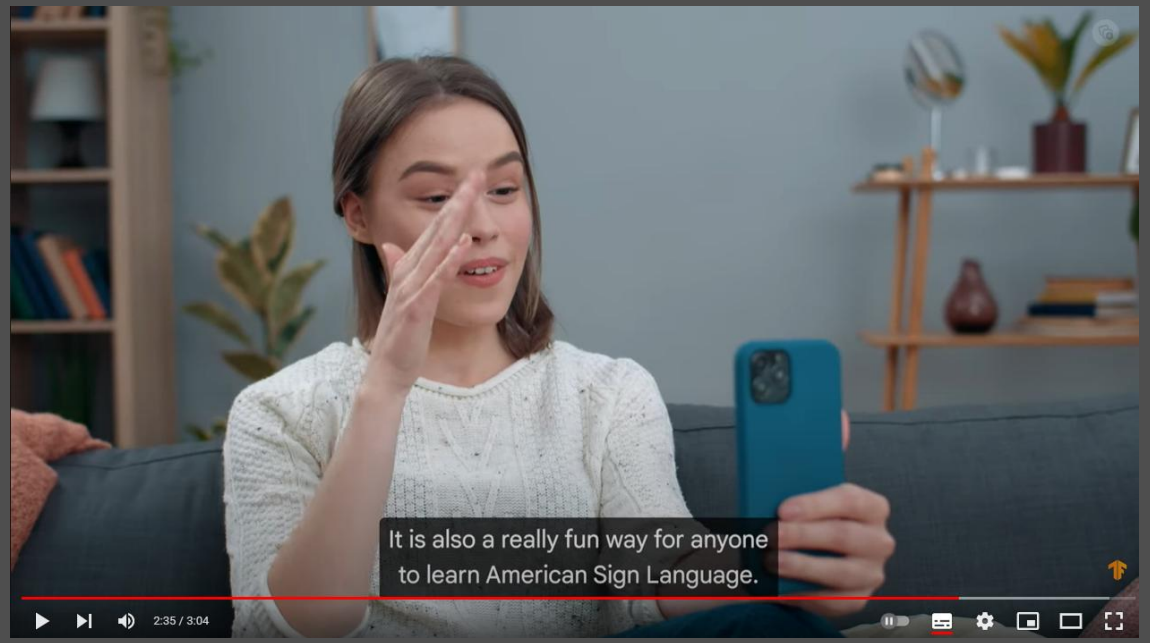
- **モデル提出**
  Androidでのリアルタイム推論のため,
  Edge AIに最適化されたTensorFlow Lite model形式へconvert
  （**推論時間：100 msec/data, モデルサイズ：40 MB以下の制約**）
  TensorFlowしか基本使えない. (scikit-learnなどのライブラリはNG)

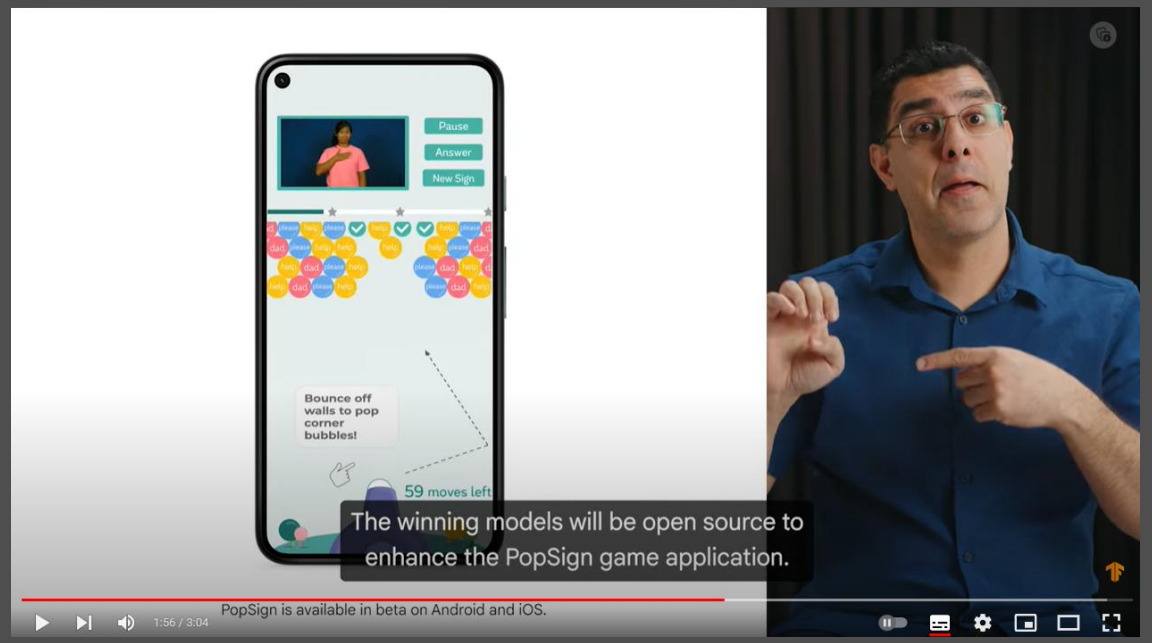
ASL例
“Listen”

- **モデルの評価**
  学習データにない被験者のSign Languageシーケンスに対しての分類精度（Accuracy）
  見えないtestデータ（Private）でのスコアで最終順位が決まる.

- **主催＆期間**
  Google, 2カ月

Google - Isolated Sign Language Recognition | Kaggle
Listen ASL - YouTube





[Join us for the first Sign Language AI Competition on Kaggle - YouTube]

コンペ概要

■ データ生成過程

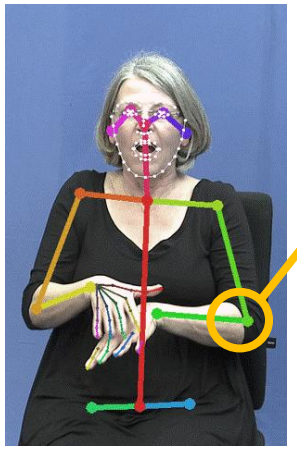
- カメラ動画 → Keypoints抽出（姿勢推定）→ 時系列座標
- 行動認識の分野(Pose-based action recognition)では一般的に行われるプロセス.
  生の動画だと画像×時間分あるので, 特徴点のみピックアップして次元削減しているともいえる.

例）1,220万画素, 30FPSだと1秒で生動画が1.1GB

Google Pixel 4a
1,220万画素

movie

Pose estimation



左肘関節のx-y-z座標
が**推定**されている.

Keypoints

Data 圧縮

提供されるデータ

次元（x, y, z）

時間

特徴量（関節座標）

3D Tensor

説明変数

対応

Look

Label

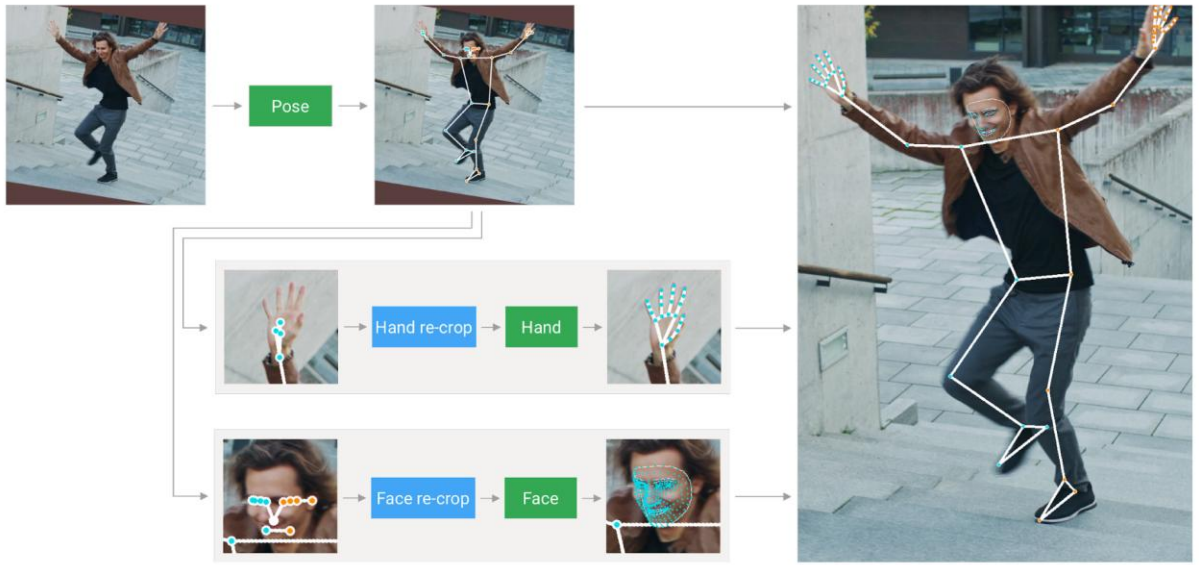
目的変数

関節座標の時間変化（多次元時系列データ）をモデリングする

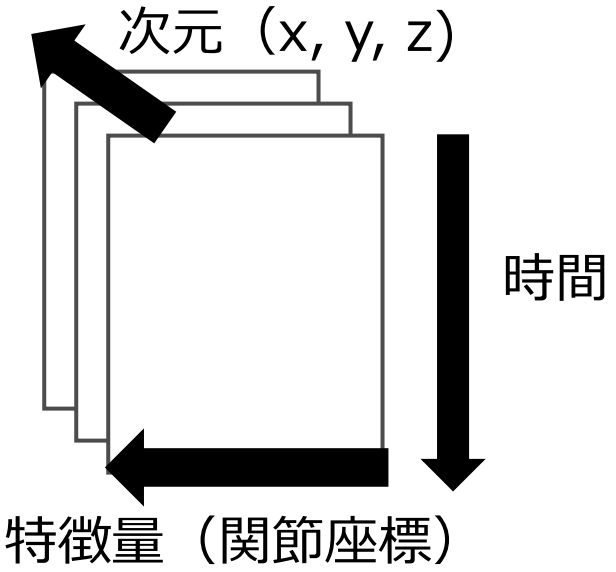
コンペ概要

■ **学習用データ**

- 被験者は21人（**スコアが算出される未知のテストデータは未知の被験者**）
- 1シーケンスデータは1単語のラベルが対応
- データ数94,477シーケンス, 250種類のラベル
- シーケンスデータは, Google Pixel 4で録画され, MediaPipe holistic modelにより,
  姿勢推定された座標として存在する.
  MediaPipe holistic modelは画像中の各関節の(x, y, z)座標を推論.
  よって, 1シーケンスデータは（時間, 特徴量, 次元）として存在する.



MediaPipe holistic model



1シーケンスデータの形状

mediapipe/holistic.md at master · google/mediapipe · GitHub

私の取り組み方

■ **My motivation**

- 得意なお題であるはずなので真剣に取り組んでメダルが取れるか実力試し.
  業務でほぼ同様のことをやっているので, そこで得たノウハウがどれくらい活きるか？

- 頑健・リアルタイム・精度を両立したモデルの作成方法を学びたい.

■ **取り組み姿勢**

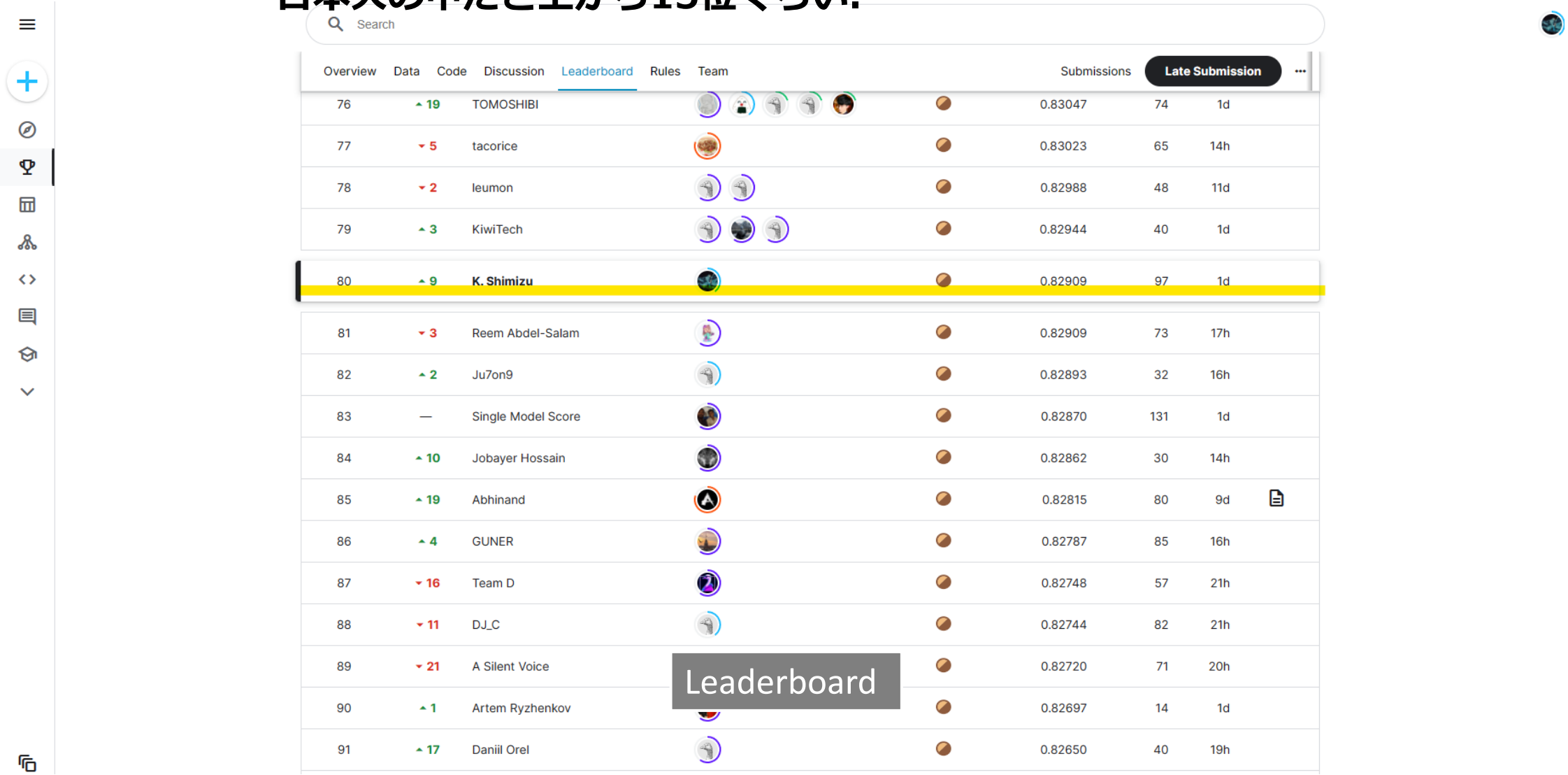
- 前半1カ月は解法のヒントになるような公開notebookやdiscussionを極力見ずに
  自力でモデルを作成する.

- 理論にあまり縛られず, “自分ならこうする”という工夫点を常に考え,
  思いついたらすぐ実装・実験して結果を確認する.（素人発想・玄人実行）

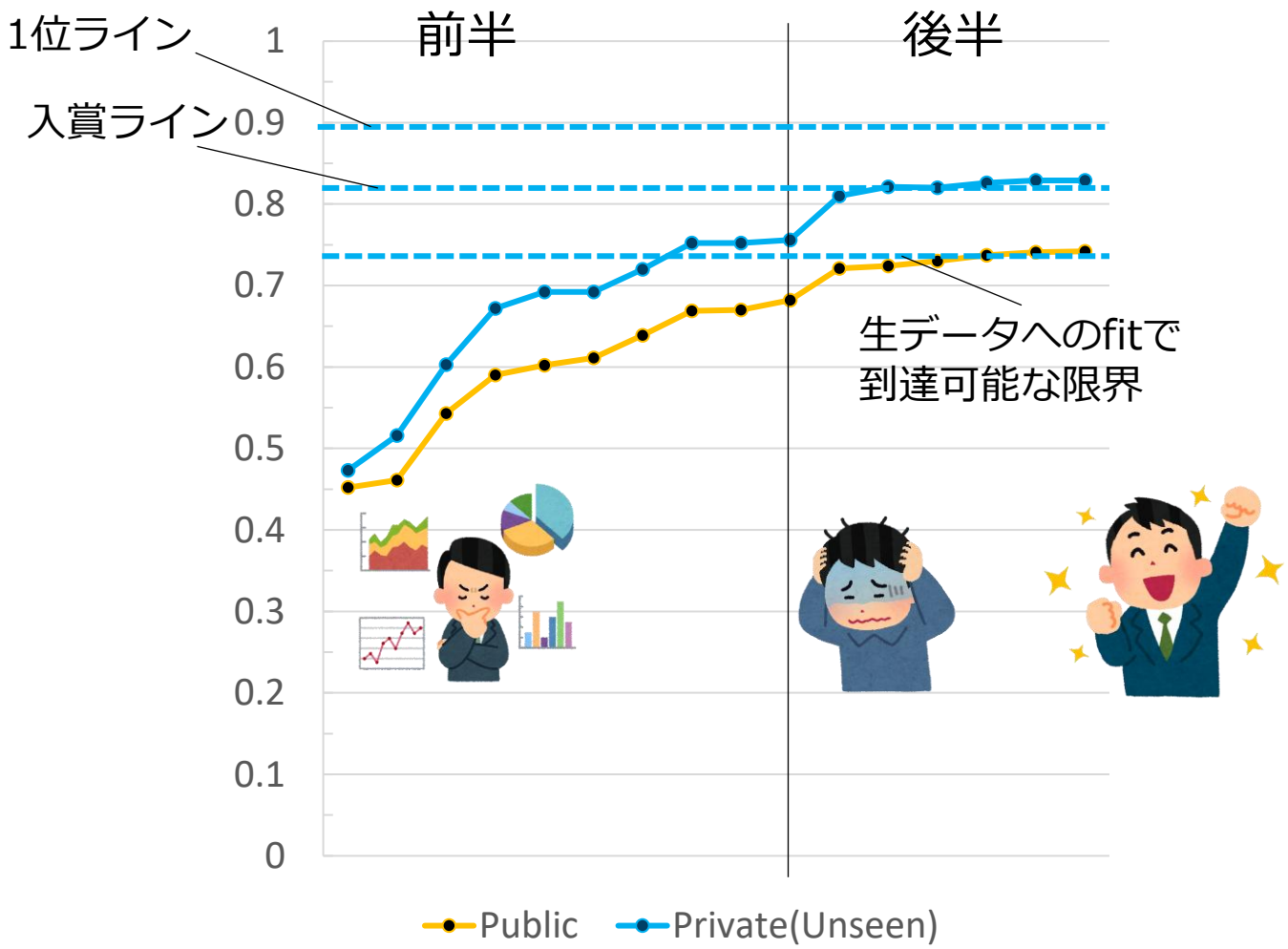
- Kaggle上でのテクニックではなく, なるべく一般的に通用することを学ぶ.

結果

1165チーム中80位で銅メダル🥉（上位10%以内が入賞範囲）
日本人の中だと上から15位くらい.



Bestスコアの変遷



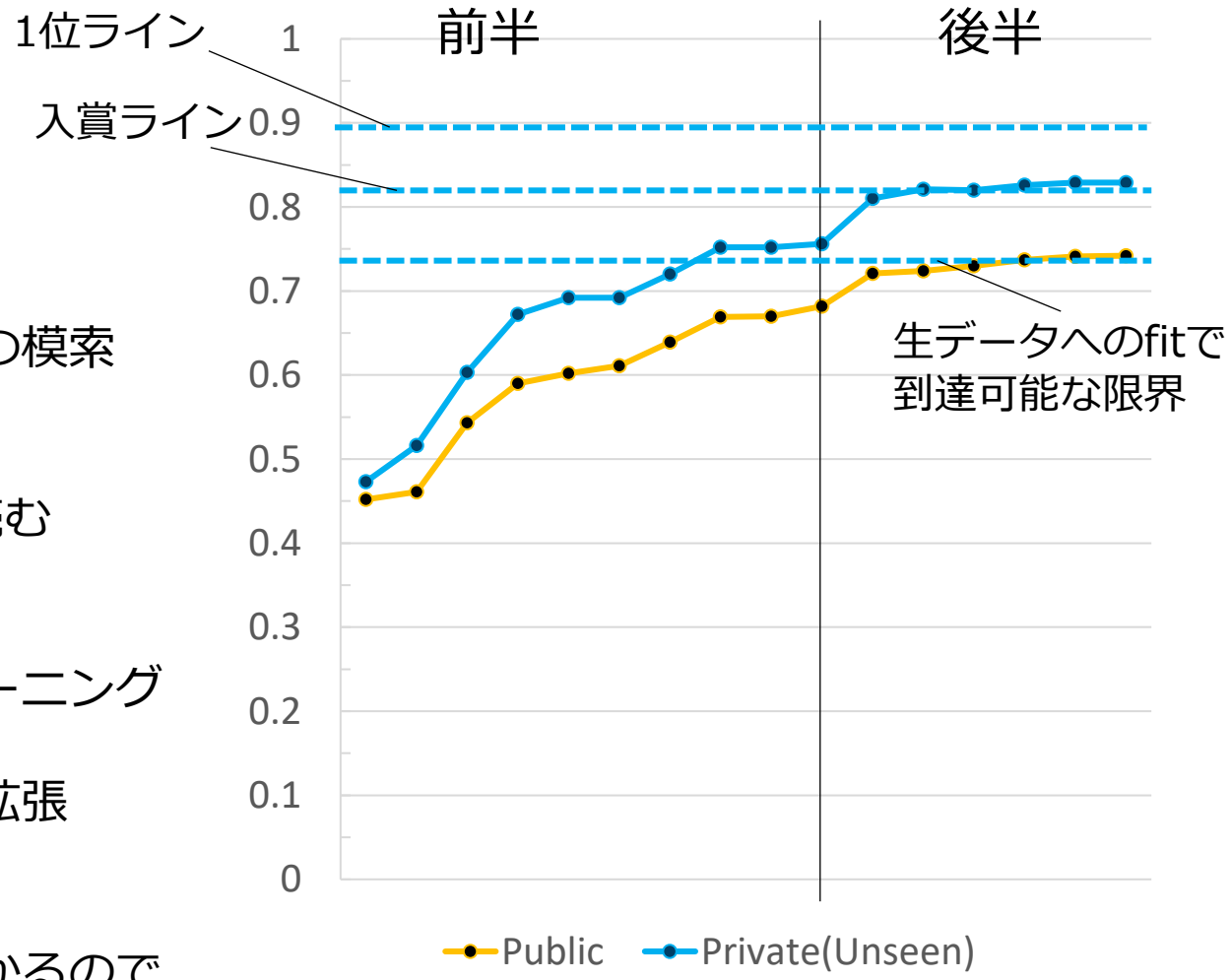
Bestスコアの変遷

スケジュール

■ **やったこと**

➢ 前半１カ月
- コンペのルール理解&環境構築
- データの可視化
- ドメイン知識の獲得
- 2層の全結合NNのモデル作成でテスト提出
- より良い前処理＆特徴量エンジニアリングの模索

➢ 後半１カ月
- スコアが高めの公開コード, discussionを読む
- 既存手法について大雑把にリサーチ
- Transformerベース＋
  特徴量エンジニアリング＆パラメータチューニング
- 全結合NNとTransformerのアンサンブル
- 回転行列やガウシアンノイズによるデータ拡張



Bestスコアの変遷

■ **生活リズム**

深層学習のクロスバリデーションには最低数時間かかるので,
寝るまでにコードを修正し寝ている間に実験を回して朝一で結果を確認する.
合計で150回くらいは実験した.

ルール理解＆環境構築

■ **ルール理解**

- 推論速度やモデルサイズに制限あり
  >最新の論文で実装されている最もよいモデルを使ったり,
  複数モデルのアンサンブルを行ったりするのは厳しそう..**人間で特徴を設計したほうがよさそう？**

- データが人間の動きに由来している
  >人によって癖があったりしそうなので, **モデルの頑健性が重要になりそうだ..**

- データのアノテーションが機械学習(pose estimation)によって行われている
  >**“計測”ではなく”推論”なのでデータをすべて鵜呑みにするのは危険な気が..特にz座標..**

- スコアは学習データにない未知の被験者のデータで計算される
  >クロスバリデーションはランダムにするのはNGで, **Group Foldクロスバリデーション**が必要だ!

■ **環境構築**

- ノートPC（GPU Docker）, クラウド（Google Colab Pro）, Kaggle notebookを併用.
  ノートPCのLaptop GPUで計算が正しく回る動作確認後, クラウドやKaggle notebook GPUで学習させる.
- **Chat GPTでTensorFlowの演算コードをある程度生成し効率化.**
- 無料枠で深層学習のクロスバリデーション回すと, 最大12時間かかるので
  Google Colab Proを契約した（A100を使うと計算時間はなんと1/6になった..!!）

データの可視化

■ **テーブルデータの動画化**

仮説を創出する上で一番やってよかった.

利き手と反対の手は使わない, 欠損がかなりある
意外と口が動く, 指の動きが重要, 顔と手の位置関係が重要,
録画開始と終了時はラベルに無関係な動作が入っている
など多くの気づき.

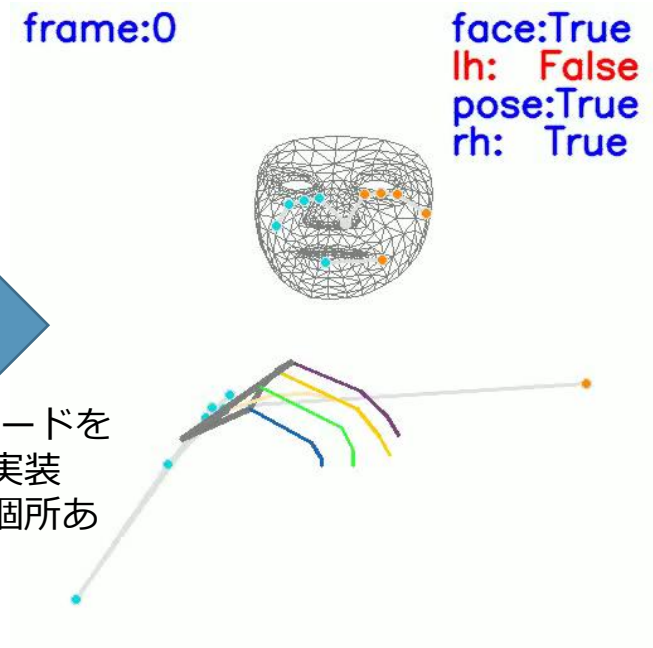
		face						
		0	1	2	3	4	5	6
		0	**1**	**2**	**3**	**4**	**5**	**6**
5	**0**	0.494	0.496	0.501	0.49	0.495	0.496	0.
6	**1**	0.501	0.493	0.498	0.488	0.492	0.494	
7	**2**	0.498	0.492	0.498	0.487	0.491	0.493	
8	**3**	0.506	0.497	0.502	0.492	0.496	0.498	0.
9	**4**	0.508	0.503	0.508	0.498	0.502	0.504	0
10	**5**	0.512	0.506	0.512	0.502	0.505	0.508	0.
11	**6**	0.52	0.512	0.517	0.508	0.511	0.514	0.
12	**7**	0.527	0.515	0.52	0.513	0.516	0.518	0.
13	**8**	0.534	0.524	0.528	0.52	0.525	0.527	0.
14	**9**	0.538	0.535	0.537	0.529	0.535	0.537	0.
15	**10**	0.539	0.539	0.541	0.533	0.54	0.541	0.
16	**11**	0.54	0.54	0.542	0.534	0.54	0.542	0.
17	**12**	0.54	0.54	0.542	0.535	0.541	0.542	0.
18	**13**	0.539	0.541	0.543	0.535	0.541	0.543	0.
19	**14**	0.541	0.541	0.543	0.536	0.542	0.543	0.
20	**15**	0.54	0.541	0.543	0.535	0.542	0.543	0.
21	**16**	0.539	0.54	0.542	0.534	0.541	0.542	0.
22	**17**	0.537	0.539	0.541	0.533	0.54	0.541	0.
23	**18**	0.538	0.539	0.541	0.533	0.539	0.541	0.
24	**19**	0.536	0.537	0.539	0.531	0.538	0.539	0.
25	**20**	0.536	0.537	0.539	0.531	0.537	0.539	0.
26	**21**	0.536	0.535	0.537	0.529	0.535	0.537	0.
27	**22**	0.536	0.536	0.538	0.53	0.536	0.537	0.
28								
29								
30								

x | y | z | +

提供データ

動画化

Googleのコードを
読み解いて実装
意外と汚い個所あり

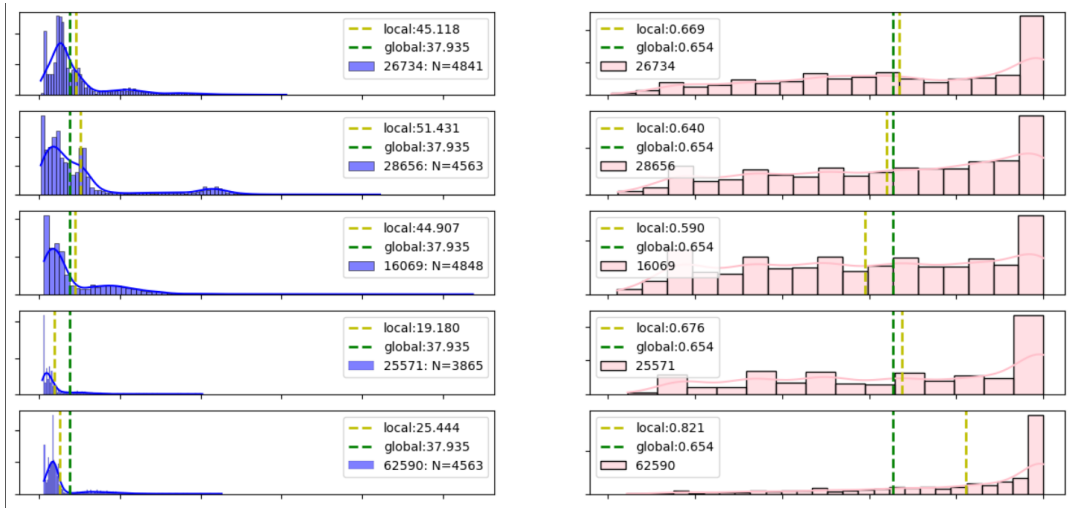


復元した
元動画

■ **被験者による動画時間, 欠損値の違いの確認**

- 動画時間
  人によって長さに違いあり.
  同じラベルでも繰り返しやったり,
  1回しかやらなかったりしてばらつく？

- 欠損値
  人によって欠損値率に違いあり.
  画角から手がはみ出したりしている.



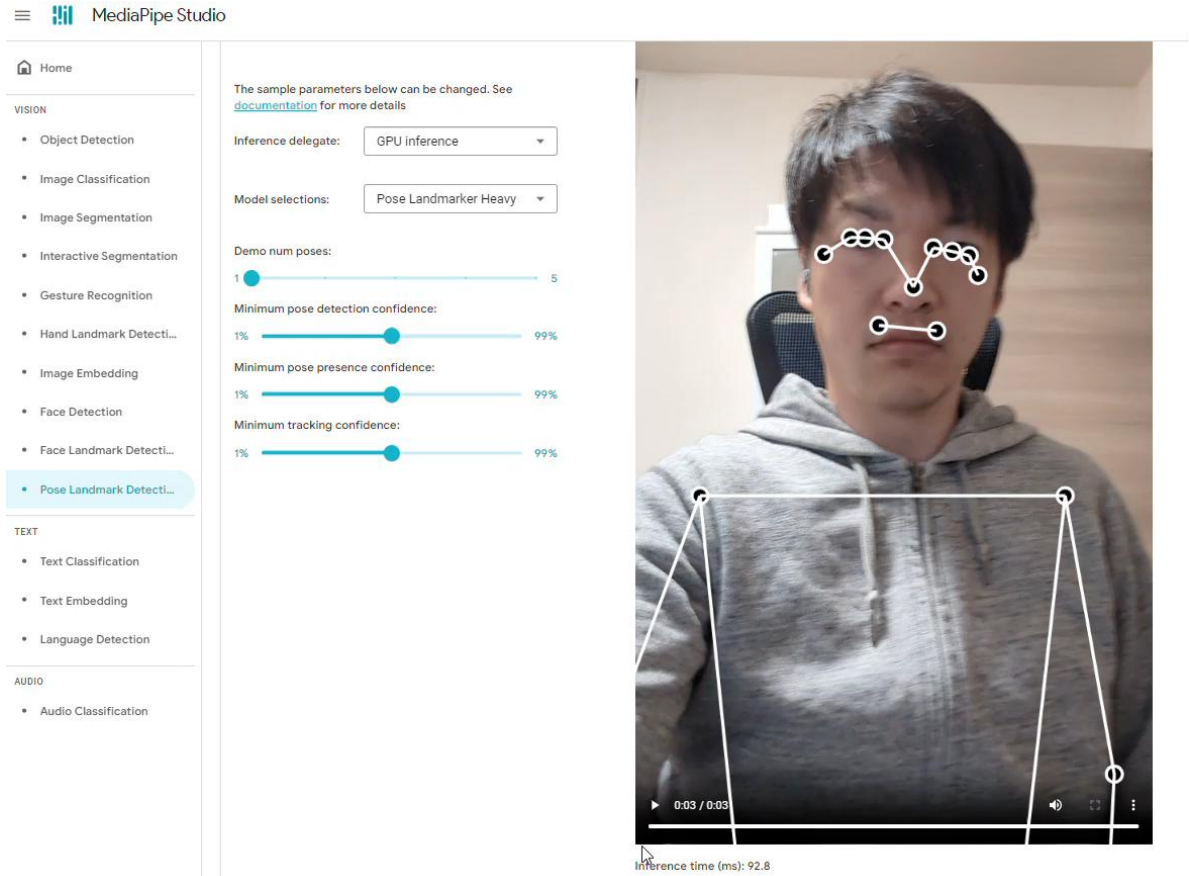
被験者ごとのデータのばらつき
（左）動画時間（右）欠損率

ドメイン知識の獲得

■ **データ取り環境の理解**
被験者のつもりでデータをとってみる.

わかったこと.

- 広角でも意外と画角狭い.
- 利き手でないほうはほぼ映らない.
  ⇒　特徴量選択
- 手持ちでブレが発生する.
  ⇒　ガウシアンノイズのデータ拡張
- 顔や肩の座標が大きく変化することはない.
- 肩幅はカメラとの距離が変わらない限り
  変化しない.
  ⇒　肩幅を距離の基準にする
- 録画開始直後と録画終了直前は,
  無意味な動作が入る.
  ⇒　時系列の最初の方と最後の方は落とす.
- 1度の動作はそんなに長くならない.



自分のGoogle Pixelで撮った動画で姿勢推定

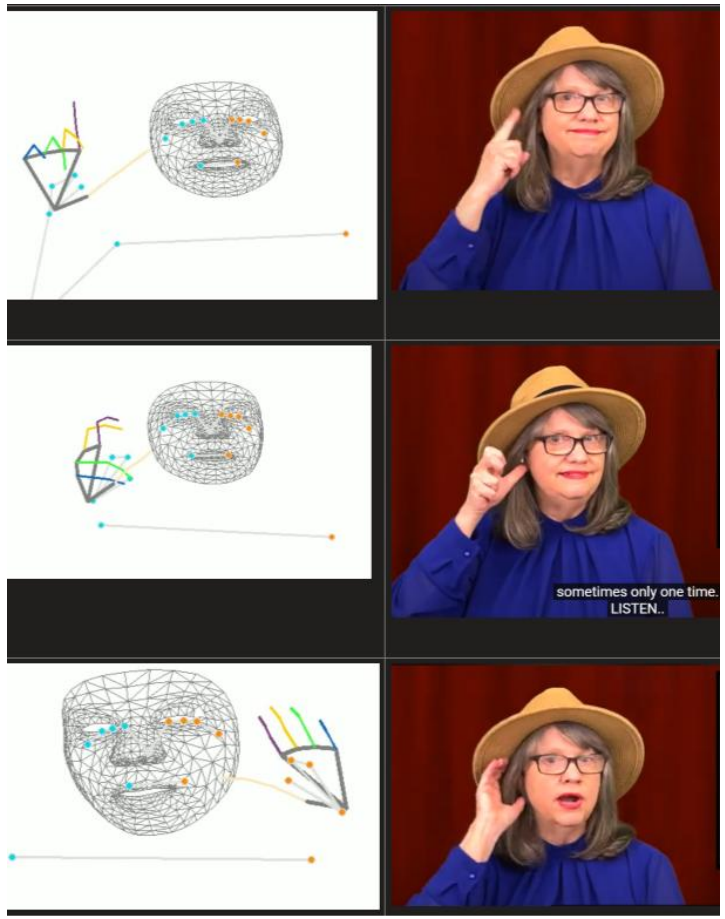
[MediaPipe Studio (google.com)]

ドメイン知識の獲得＆特徴量エンジニアリング

■ **ASLの動作の理解**
ASLのYouTubeを見て, 一般的にどのような動作があるか確認.
可視化で確認した通り, **指の動き, 手と顔の位置関係, 口の動きが重要そうだ..**

■ **前処理＆特徴量創出**
素人仮説で様々な特徴量を作成し, 2層のNNでクロスバリデーションをして
どのような特徴量が効くかを検証（20パターンくらい実験した.）
精度向上＆頑健性の確保＆学習収束の高速化＆推論速度の向上を目指す.

- フィルタリング
  利き手が映っているフレームのみ使用
- 生座標
  z座標は不要.
  唇, 肩&肘, 利き手だけあれば十分（特徴量削減で推論速度向上）
  利き手は動画内に映り込みが多いほう（欠損が少ないほう）とした.
- 距離
  手と顔の距離, 手首と各指先の距離
- 角度
  手の指の各角度, 手-ひじ-肩の角度（腕の曲がり具合を見たい）
- 速度＆加速度
  生座標に関して計算. 大きく動く関節が手話に関係しているはず.



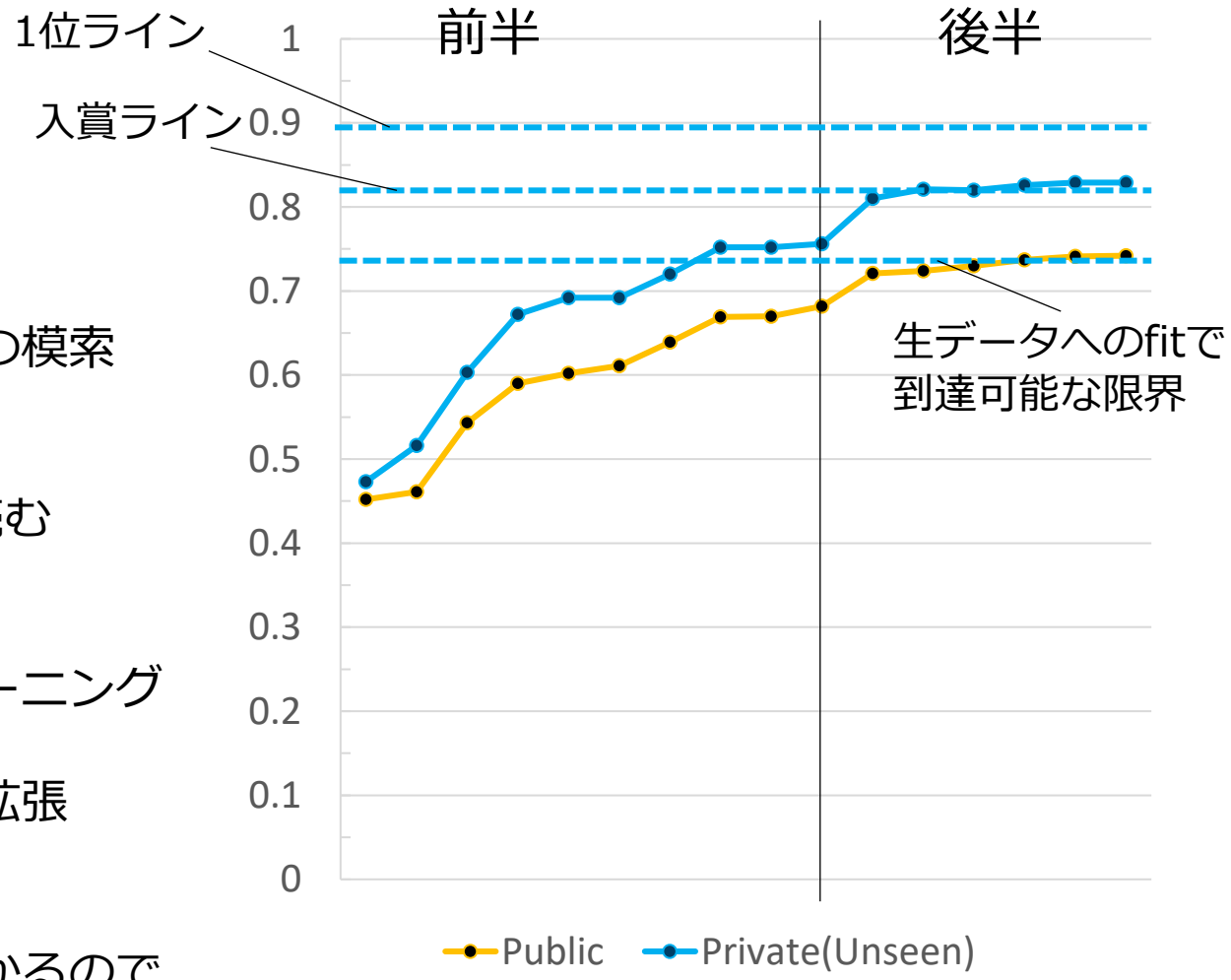
提供データとYouTubeの比較

スケジュール

■ **やったこと**

➢ 前半１カ月
- コンペのルール理解&環境構築
- データの可視化
- ドメイン知識の獲得
- 2層の全結合NNのモデル作成でテスト提出
- より良い前処理＆特徴量エンジニアリングの模索

➢ 後半１カ月
- スコアが高めの公開コード, discussionを読む
- 既存手法について大雑把にリサーチ
- Transformerベース＋
  特徴量エンジニアリング＆パラメータチューニング
- 全結合NNとTransformerのアンサンブル
- 回転行列やガウシアンノイズによるデータ拡張



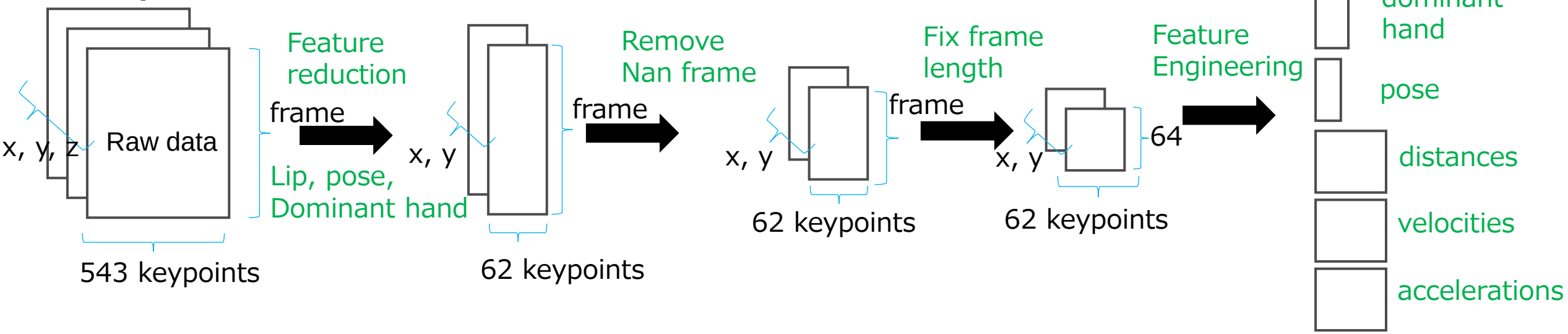
Bestスコアの変遷

■ **生活リズム**

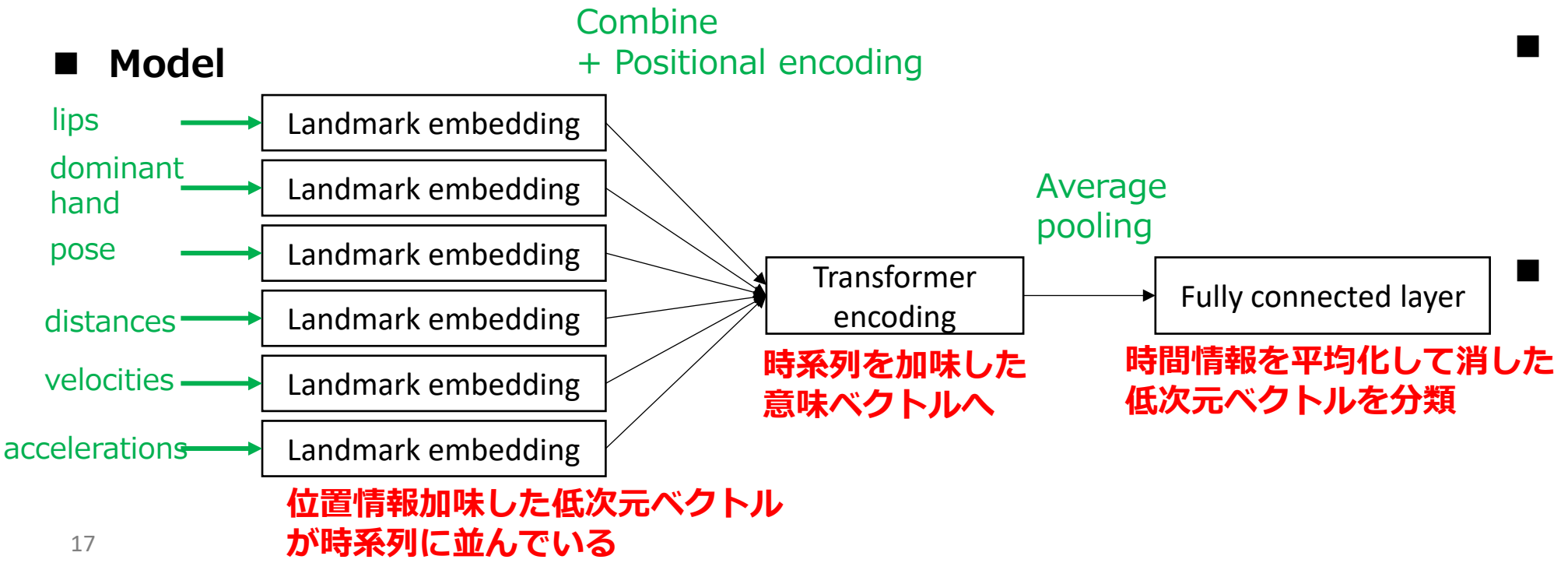
深層学習のクロスバリデーションには最低数時間かかるので,
寝るまでにコードを修正し寝ている間に実験を回して朝一で結果を確認する.
合計で150回くらいは実験した.

My solution overview

■ Preprocess



■ Model



■ Regularization

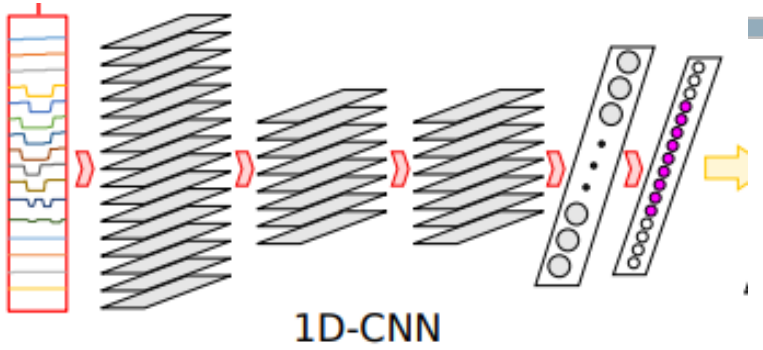
- Dropout
- Label smoothing
- Weight decay

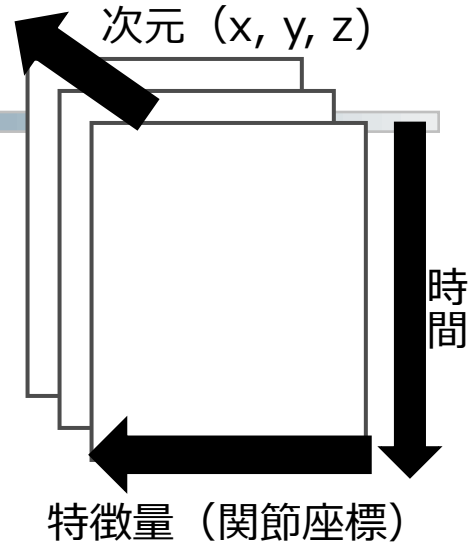
■ Data augmentation

- Gaussian noise
- Random rotation

他の参加者のsolutionを見て..

- **時系列データへのCNNの適用.**
  CNNで時間方向に畳み込み1D CNN.
  データを3チャンネルの画像とみなす.
  （音データをスペクトログラム→画像認識と同じ）
  CNNは基本的にRNNよりも推論速度が速い.（？





- **圧倒的にTry&Errorでスコアを上げている**
  私の2倍, 3倍, ..の量の実験を平気で回している.
  48h/dayでGPUを回した強者..
  計算リソースの確保, 実験管理ツール導入, 最小限の変更で実験可能なコードの柔軟性.
  仮説だしのためにYouTube見る, データ取り取得ツールを作って自分の手話をとってみる..

- **データ拡張, 特徴量エンジニアリングが重要だった.**
  生座標の学習で何も工夫しないとほぼ確実に過学習する.
  生座標のノイズに影響しないロバストな特徴量（距離, 角度, 速度, 加速度…）
  データ拡張にも様々な工夫が見られた.（ランダム回転, ランダムフレーム欠損, ランダムリサイズ…）

感想

Kaggleでメダル圏内に入るには..

- データの分割～学習・推論までのベースをなるべく早めに作る
- 公開されている情報はなるべく読み込む（code, discussion..）
- 問題背景を読み込み, 特徴量やモデルなどの仮説をいろいろ立てる
- 思いついたことはすべてやりきるレベルで実験（思い込みでアイディアを検証せずに捨てない）

個人的には, 公開されている情報＋5個くらいの自前アイディア（公開ノートブック+3~5%）で銅メダル圏内に入れると思います.

データが重い, 膨大な計算環境が必要など参入障壁が高いコンペほど, 少しの改善で銅メダル圏内に入れると思います.