
ベンチャー体験工房2 後期 最終発表
2023/2/6

たんぽぽ画像分類

s1290162 中村善音

たんぽぽ1計画とは

・国際ステーション上で行われている日本初のアストロバイオロジー実験。

・パンスペルミア説を検証する

・エアロゲル(低密度ガラスの捕獲材)を用いて**宇宙塵**(micrometeoroid)などを捕集する(捕集実験)。

→その物質を地上で分析する

- **宇宙塵**(**宇宙由来**の有機物含有物質)

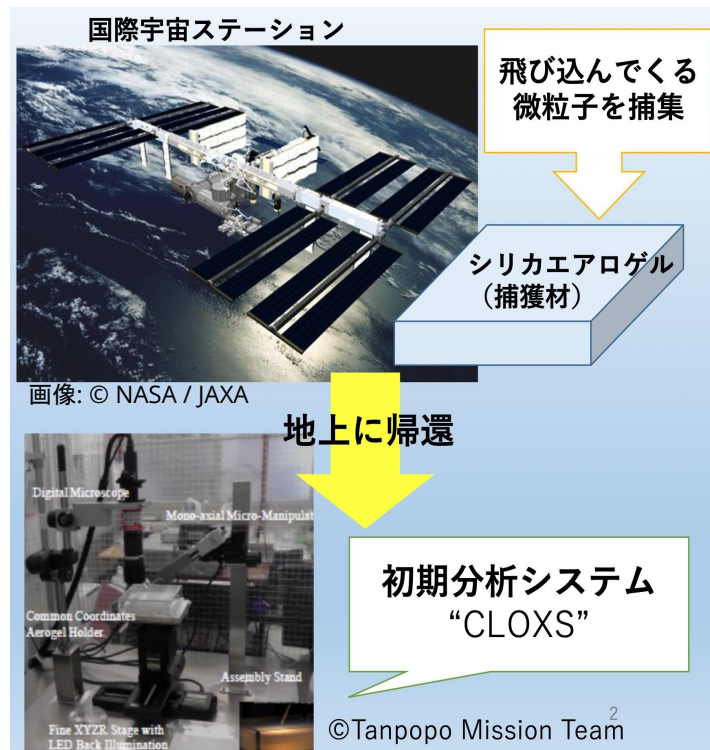
→ 1mm以下の天然の微粒子

- **地球由来の微粒子**(**地球由来**の微生物含有物質)

- スペースデブリ(人工物)



画像: © Tanpopo Mission Team

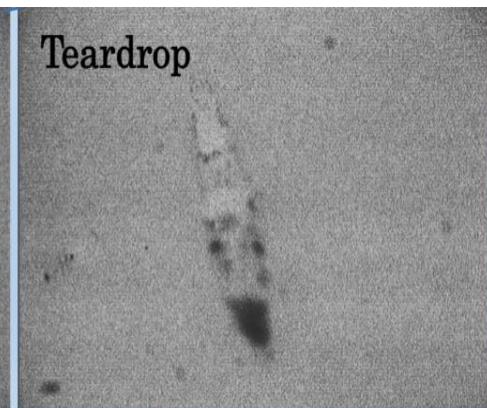
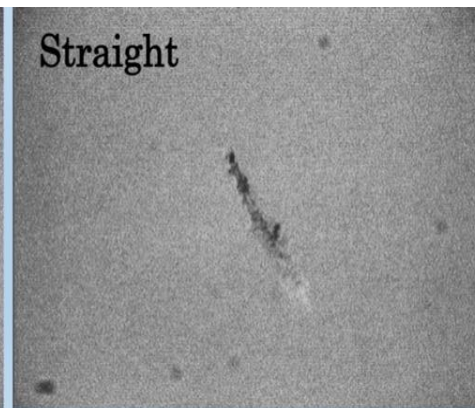
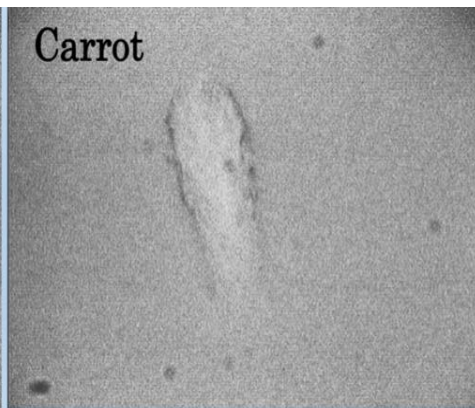
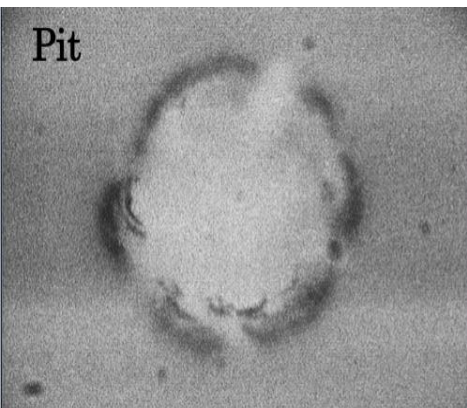


捕集されたトラック (貫入孔) の分類

- トラック (微粒子貫入孔・衝突痕)

Pit, Carrot, Straight, Teardrop

ゲル中に貫入しており、深さを持つ (= 画像枚数が多い)

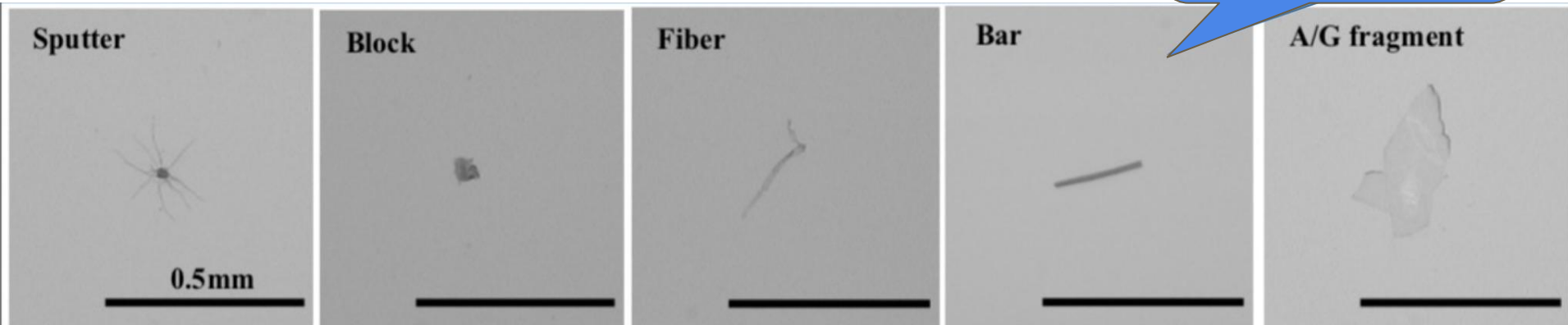


補修された表面付着物の分類

- 表面付着物(主にゴミ)

Sputter, Block, Fiber, Bar, A/G fragment

表面のみで、深さはない
(=画像枚数は少ない)



地上初期分析の流れ -CLOXSによる画像撮影

パネル表面の顕微画像の自動撮像・統合・マッピング

貫入孔候補
や表面付着物等の検出
・ID付与

貫入孔候補
への再訪問
と拡大・三
次元撮像

研究所による貫入孔の判定・分類・掘削用立体形状の決定

試料配分委員会
による詳細分析
チームへの配分
決定

試料の自動
切り出し

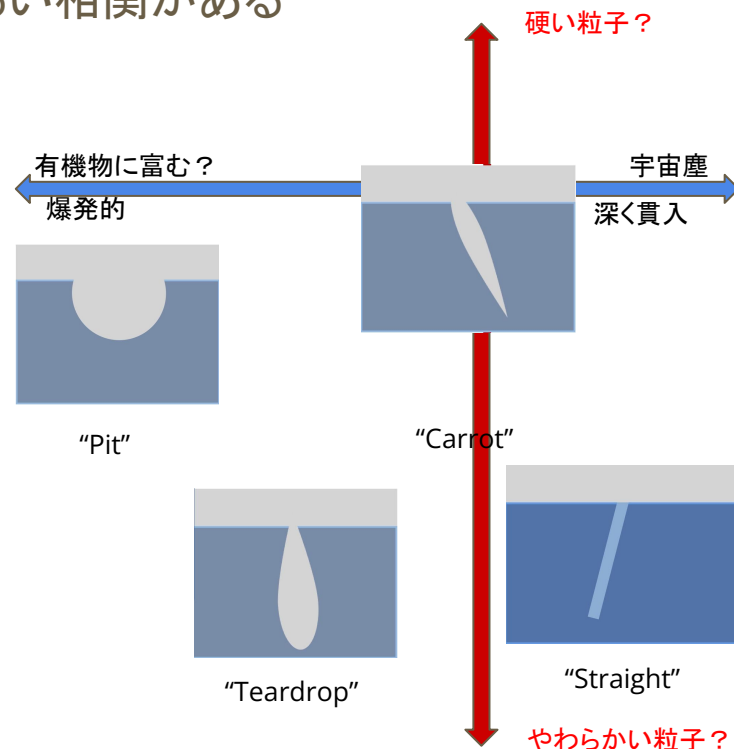
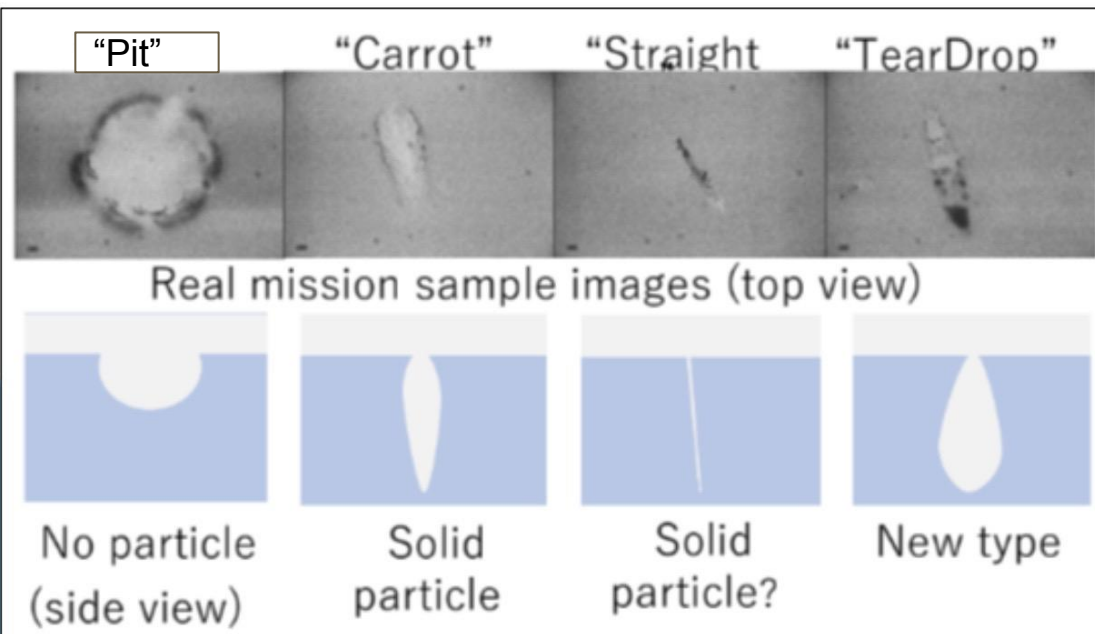
CLOXSの一連の流れ

貫入孔画像や付随する各種データをデータベースに格納、各分析チームへ分配

タイプ分類と捕獲粒子の物性推定

- トラック(貫入孔)の形状と捕獲物の物性とはゆるい相関がある

画像: © Tanpopo Mission



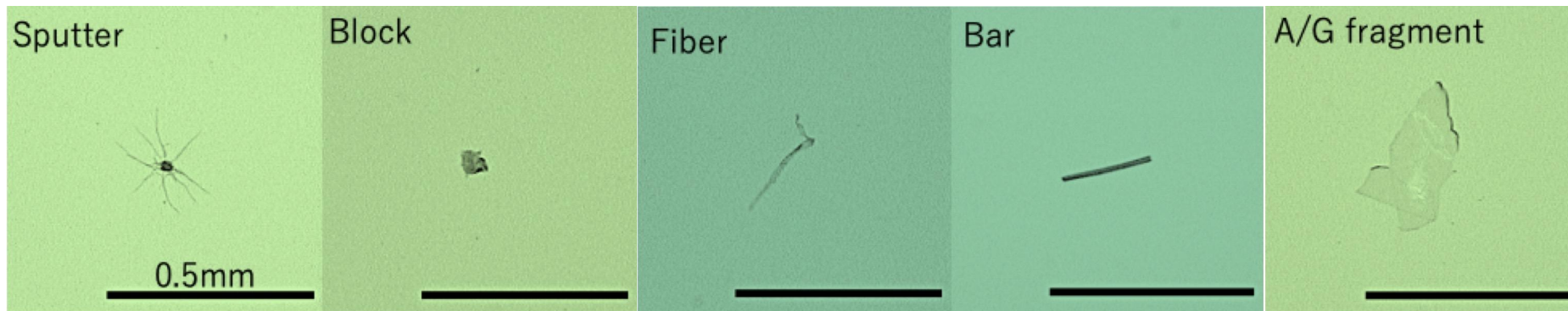
表面付着物の分類

表面付着物の分類

- 桐生さんの論文

[1] <https://web-int.u-aizu.ac.jp/thesis/Thesis2020-Mar/s1250153/s1250153.pdf>

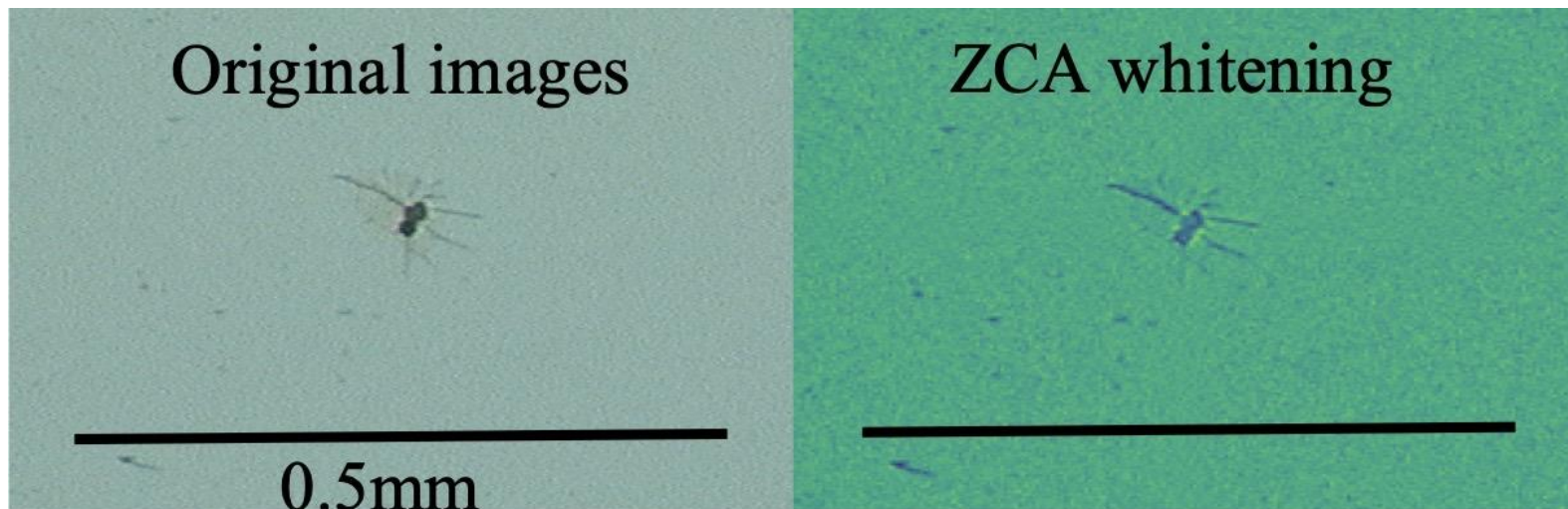
- 表面付着物を自動分類したときに、再現率(recall)が70%以上を目標する



データの前処理 (Preprocessing)

- ZCA Whitening

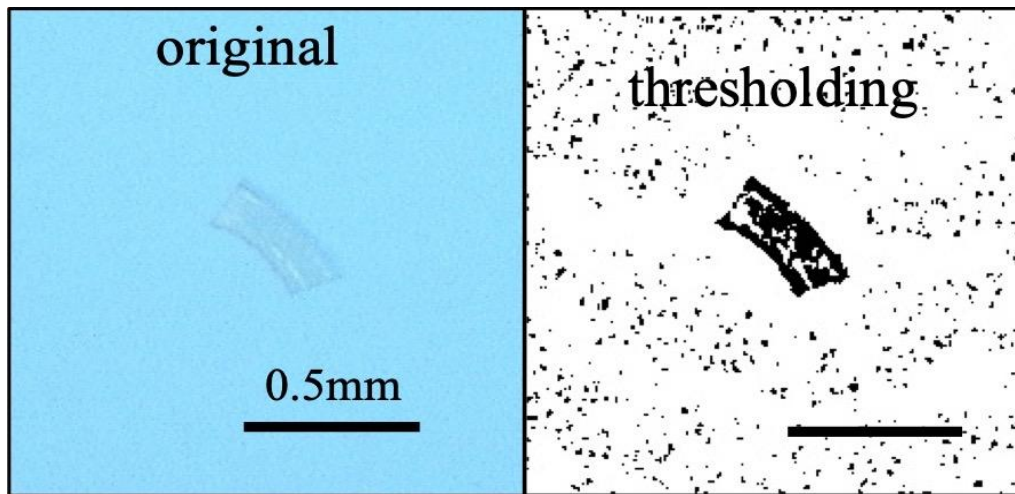
白色化: トレーニングデータの偏りを小さく、また各成分間の相関を無くす



データの前処理

- Thresholding(しきい値処理)

特定のしきい値以上(もしくは以下)のスコアを持つピクセルだけを抽出する手法



©Tanpopo1 Mission Team

表面付着物の分類

前処理(Preprocessing)

- ・リサイズ Resize
- ・切り取り CenterCrop

データ拡張(Online data Augmentation)

- ・左右回転 RandomRotation
- ・上下左右に反転 RandomVerticalFlip, RandomHorizontalFlip
- ・明るさ、コントラスト、彩度、色相を変化 ColorJitter

表面付着物の分類

ネットワーク(Deep Neural Network)

- ・VGG16 に **Batch Normalization** を加える

- 分布の偏りを失くして、学習を安定させる？

- 学習率(Learning Rate)を大きくできる

- 局所 min に陥らない

[2] <https://proceedings.neurips.cc/paper/2018/file/36072923bfc3cf47745d704feb489480-Paper.pdf>

- ・**Vision Transformer**

- 計算コスト削減？

Batch Normalization

ミニバッチごとにネットワークのレイヤへの入力の分布が変わる

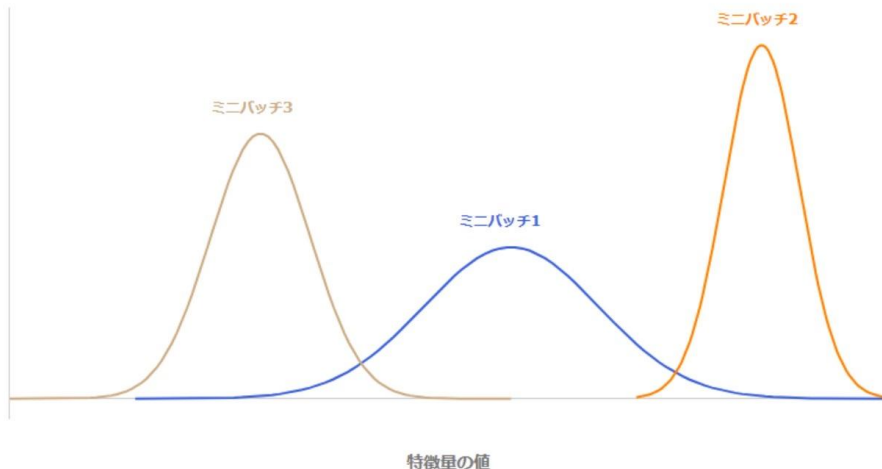
→ 学習が安定しない

各レイヤごとに入力を正規化



ミニバッチごとの平均・分散を使って

ミニバッチごとに正規化



VGG16 + Batch Normalization

- ReLU層とMaxPooling層の間にBatch Normalizationを追加(理由: 情報を圧縮する前に正規化したいから?)

(参考) [4] <https://www.nature.com/articles/s41598-021-95240-y>

- Batch Normalization層のパラメータ、後半の全結合層のパラメータを更新(勾配計算 True)
- 他のパラメータは ImageNetで学習された重みを使用(Fine Tuning)

合計パラメータ数: 134, 283, 973 (1億3千万)

訓練パラメータ数: 119, 569, 285

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (12): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (17): ReLU(inplace=True)
    (18): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (19): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (20): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (21): ReLU(inplace=True)
    (22): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (23): ReLU(inplace=True)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU(inplace=True)
    (26): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (27): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): ReLU(inplace=True)
    (32): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (33): ReLU(inplace=True)
    (34): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (35): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=5, bias=True)
  )
)
```

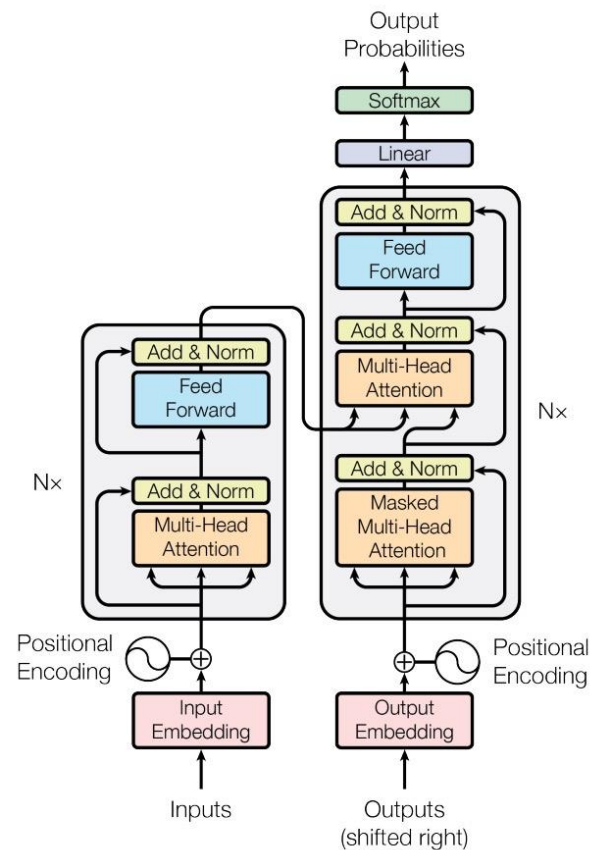
Transformer

- CNN, RNNなどを使わないで、Attention 機構のみを使う

- Attention

→ どの情報に注目すべきか判断して情報を処理

- Encoder, Decoder



[5]

<https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>

Vision Transformer (ViT)

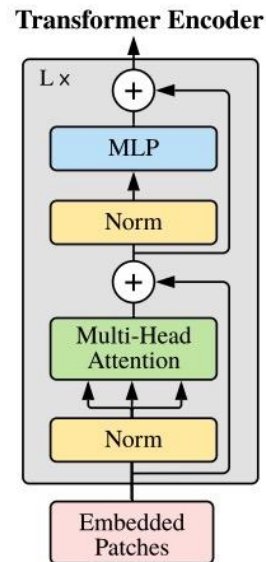
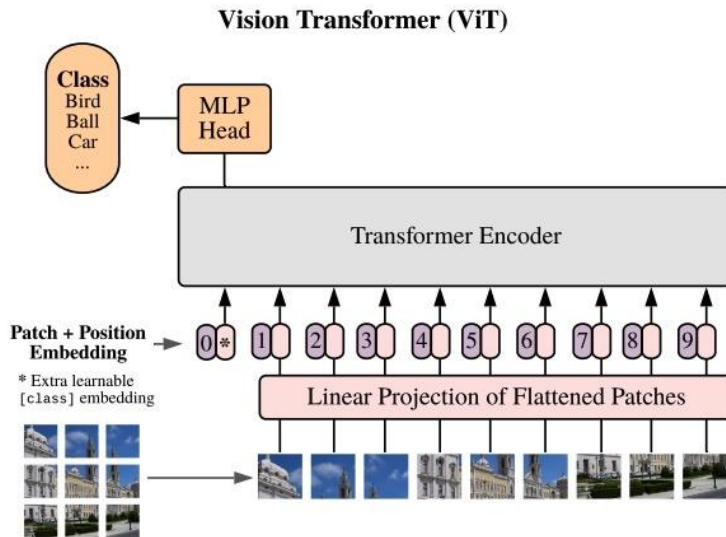
- Transformer の Encoder 部分を使う

- 画像パッチを単語のように扱う

- 入力: CLS token 画像全体の特徴を表すベクトル (分類タスクにおいて重要)

- Hugging Face

大規模データセットによる事前学習
+ファインチューニング



Vision Transformer

- ・google のImageNet(2.1万クラス)で事前学習されたモデル
(google/vit-base-patch16-224-in21k)を使用
- ・最後に分類の出力として全結合層 (nn.Linear) を追加
- ・最後の層のパラメータのみ更新 (勾配計算 True)

合計パラメータ数: 86, 393, 093 (8千万)

訓練パラメータ数: 3, 845

交差検証 (K-Cross-validation)

K = 5 Train data : Validation data = 4 : 1 = 196 : 49 (Train data = $49 \times 5 = 245$)

1 st	Validation	Train	Train	Train	Train
2 nd	Train	Validation	Train	Train	Train
3 rd	Train	Train	Validation	Train	Train
4 th	Train	Train	Train	Validation	Train
5 th	Train	Train	Train	Train	Validation

交差検証 (Cross Validation) の結果

5-Fold Accuracy

・ViT

平均 99.6 標準偏差 0.0067

・EfficientNet

平均 88.2 標準偏差 3.9573

・VGG16 + BatchNormalization

平均 100.0 標準偏差 0.0000

・VGG16

平均 93.9 標準偏差 0.1166

テスト結果 1 (各クラス 21枚ずつ)

Accuracy: 予測がどの程度当たったか

Network / Class	Sputter	Fiber	Block	Bar	A/G Fragment
Vision Transformer	99.8±0.38	97.9±0.38	97.1±0.00	99.0±0.00	96.2±0.00
VGG16 + Batch Normalization	100.0±0.00	98.1±0.00	99.8±0.38	99.0±0.00	98.9±0.38
VGG16	98.9±0.71	96.0±1.52	99.0±0.60	98.5±0.47	93.9±0.97
EfficientNet-B2	94.1±1.11	90.9±1.29	93.7±0.97	95.0±1.11	92.0±0.97

テスト結果 2

Recall: 実際にSputterであるもののうち、Sputterと予測した割合

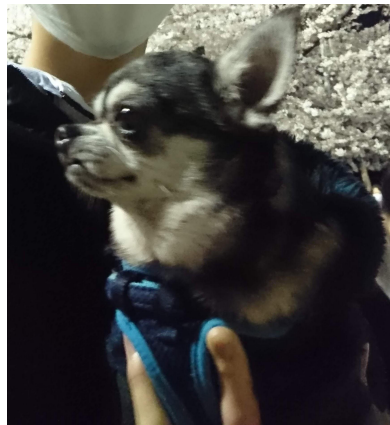
Network / Class	Sputter	Fiber	Block	Bar	A/G Fragment
Vision Transformer	100.0±0.00	90.5±1.52	95.0±0.00	100.0±0.00	90.5±0.00
VGG16 + Batch Normalization	100.0±0.00	95.2±0.00	99.1±1.82	95.5±0.00	100.0±0.0
VGG16	97.2±2.26	84.2±4.86	99.1±1.82	94.5±1.62	93.1±4.29
EfficientNet-B2	85.0±3.13	73.0±3.82	81.6±2.86	91.6±2.58	87.2±4.09

テスト結果 3

Precision: Sputterと予測したうち、実際にSputterである割合

Network / Class	Sputter	Fiber	Block	Bar	A/G Fragment
Vision Transformer	99.0±1.90	100.0±0.00	90.5±0.00	95.2±0.00	90.5±0.00
VGG16 + Batch Normalization	100.0±0.00	95.2±0.00	100.0±0.00	100.0±0.00	94.3±1.90
VGG16	97.1±3.81	99.0±1.90	96.2±3.56	98.1±2.33	75.2±3.56
EfficientNet-B2	85.7±4.26	86.7±1.90	88.6±2.33	82.9±3.81	70.5±1.90

愛犬のチワワと他のチワワの分類



130枚

188枚



議論 Discussion

愛犬のチワワと他のチワワの分類

テスト結果

・VGG16+Batch Normalization

```
Mydog
VGG16 + Batch Normalization network
['myChihuahua', 'ohterChihuahua']
[[14 0]
 [18 0]]
```

```
Accuracy : [43.8 43.8]
Recall : [43.8 nan]
Precision : [100. 0.]
```

・Vision Transformer

```
Mydog
Vision Transformer network
['myChihuahua', 'ohterChihuahua']
[[14 0]
 [ 1 17]]
```

```
Accuracy : [96.9 96.9]
Recall : [ 93.3 100. ]
Precision : [100. 94.4]
```


やること(メモ)

・表面付着物

- 本のレビュー
- 画像の枚数を増やしたら、Google Colaboratryでの実行がうまくいかない

-> プロセスのキャッシュ？

-> VGG16 モデルをVision Transformerに変える？

- Vision Transformer (Hugging Face)

VGG16 + Batch Normalization

efficientNet

・トラック

- image sequence

-> context approach (Multi View CNN, LSTM, Vision Transformer)

classification より object detection の方がいいのでは？

アンサンブル学習、画像の位置、長さのみを扱うモデルを準備

教師あり(明確分類されているものデータで) → 教師なし(中間的なデータで)

表面だけから判断(分類)できれば CLOXSのシステムとしても扱いやすい？

参考文献 References

[1] KIRYU, Honoka. "Automatic Classification of Surface Objects on Aerogel Images for the Tanpopo Mission" University of Aizu, Graduation Thesis. March, 2021

<https://web-int.u-aizu.ac.jp/thesis/Thesis2020-Mar/s1250153/s1250153.pdf>

[2] Bjorck, Nils, et al. "Understanding batch normalization." *Advances in neural information processing systems* 31 (2018).

<https://proceedings.neurips.cc/paper/2018/file/36072923bfc3cf47745d704feb489480-Paper.pdf>

[3] Batch Normalizationを理解する

<https://data-analytics.fun/2021/09/11/understanding-batch-normalization/> (参照 2022-12-12)

[4] Yang, Haoyan, et al. "A novel method for peanut variety identification and classification by Improved VGG16." *Scientific Reports* 11.1 (2021): 1-17.

<https://www.nature.com/articles/s41598-021-95240-y>

参考文献 References

[5] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

<https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>

[6] Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

<https://arxiv.org/pdf/2010.11929.pdf>

[7] Miyamoto, Y., et al. "Classification of Track Types in Tanpopo Mission by Deep Learning." *LPI Contributions* 2678 (2022): 1911.

<https://www.hou.usra.edu/meetings/lpsc2022/pdf/1911.pdf>

ご清聴ありがとうございました。