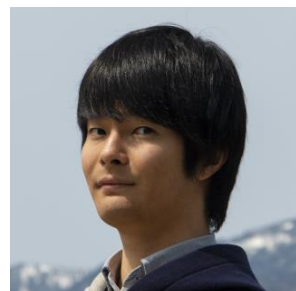


Python 始めて数か月のエンジニアが

# 画像検査自動化に挑戦

---



三協マテリアル社 技術開発統括室  
佐藤 晃太

## 建材事業



## 商業施設事業



## 国際事業

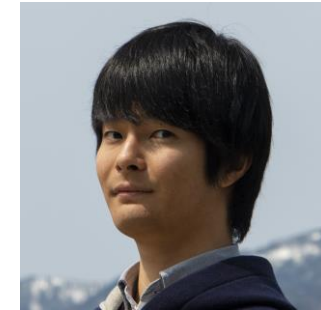


## マテリアル事業

### 素材をカタチにする

アルミニウム・マグネシウムのビレットと型材製品を提供

三協立山株式会社 <https://www.st-grp.co.jp/>



佐藤 晃太

Kota Sato - 27 歳

三協マテリアル社  
技術開発統括室

大学院修了 (化学系)  
プログラミング未経験の状態

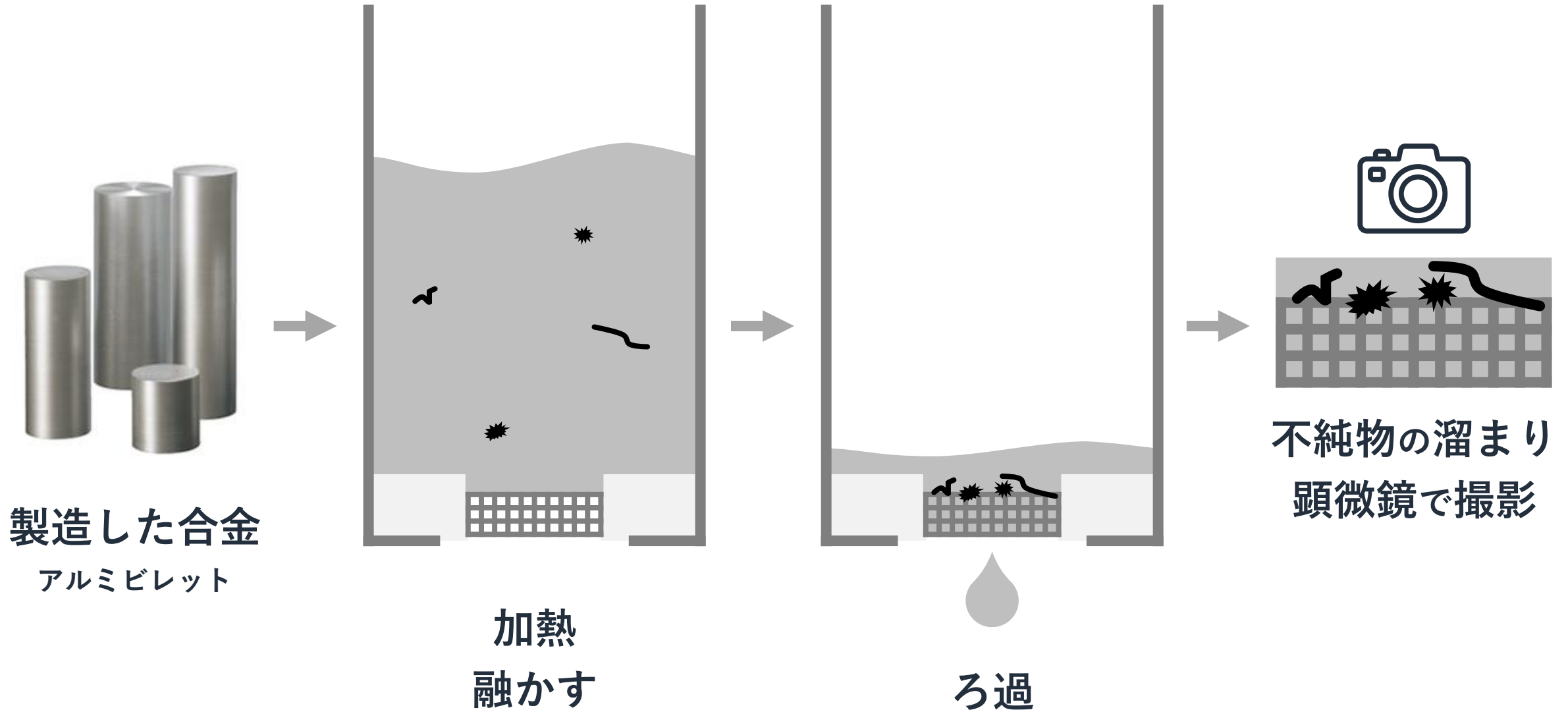


三協立山に入社  
実験試料・製品の分析業務に配属



とある画像検査に悩むことに...

# 製造した合金内の不純物を検査する



# 不純物検査 - 目視に頼った画像検査



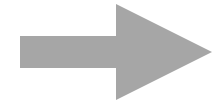
顕微鏡



熟練者

- 熟練度
- 思い込み
- 技術継承

目視で  
不純物を識別



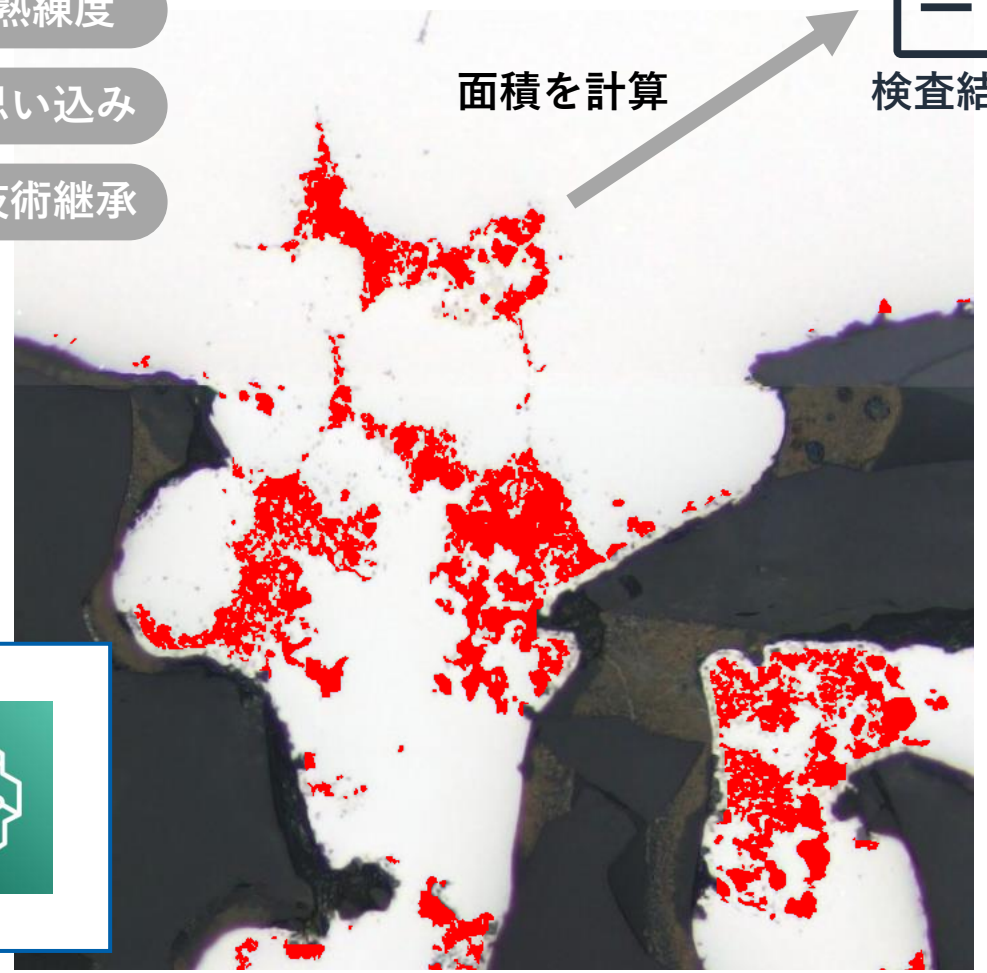
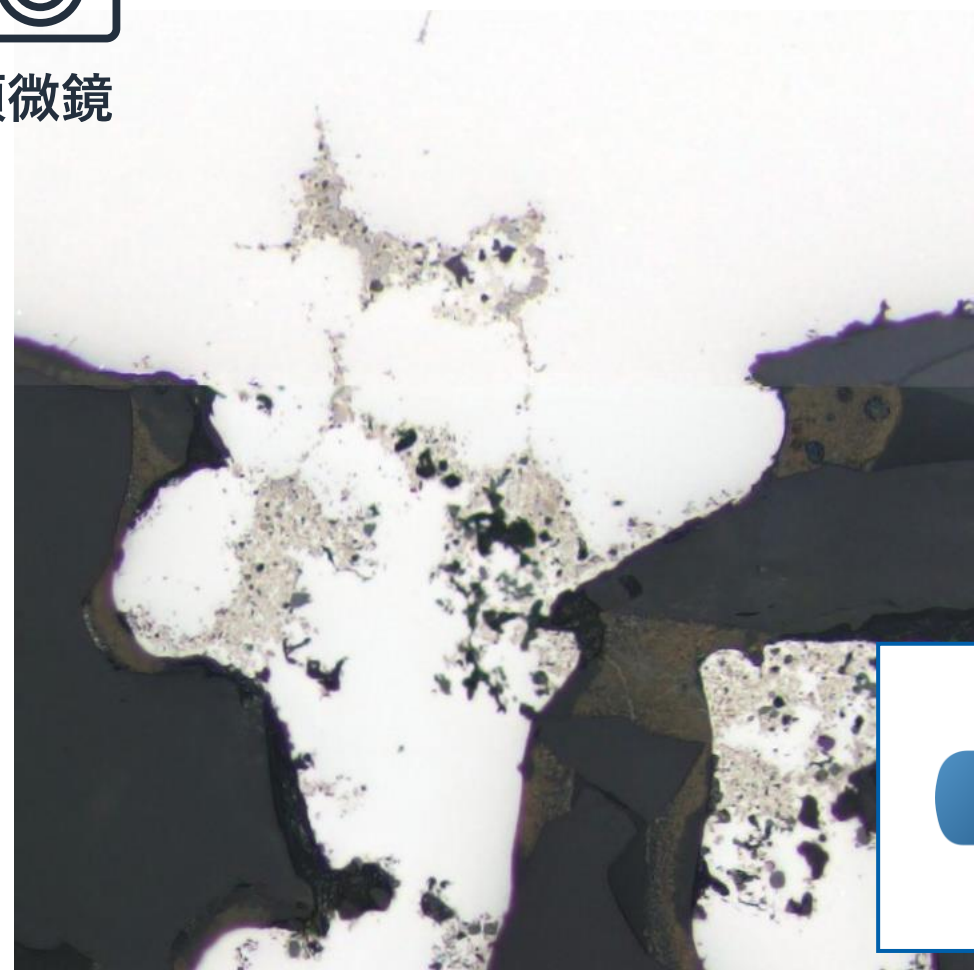
自動化



面積を計算

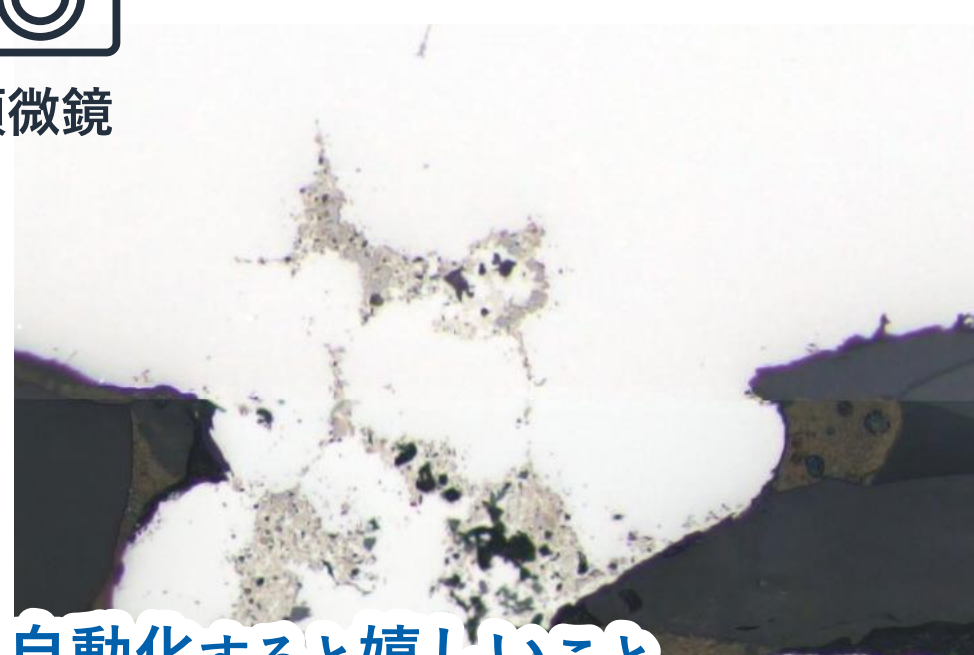


検査結果





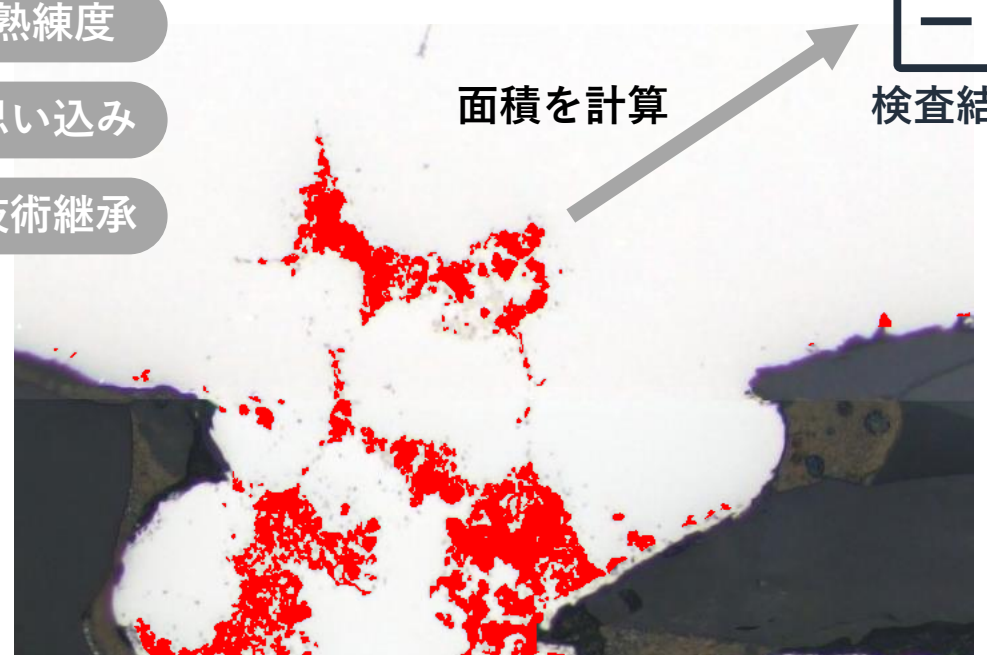
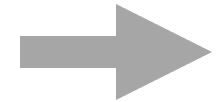
顕微鏡



熟練者

- 熟練度
- 思い込み
- 技術継承

目視で  
不純物を識別



面積を計算



検査結果

## 自動化すると嬉しいこと



作業の平準化・負荷軽減

- 技術継承が容易になる
- 作業者によるバラツキが減る

検査 1 回当たりの工数削減

- 高頻度で検査可能になる
- 品質管理の強化に繋がる

機械学習の社内事例

- 新しい改善への機運が高まる



# Python でプログラムの自作に挑戦！...挫折

初めての  python™

画像処理ができるらしい！  
やってみるぞ！



取りこぼしがある...  
プログラム自作は難しい

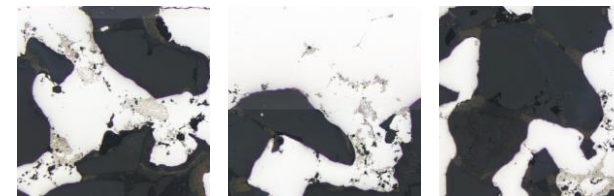


```
core.py - stalin - Visual Studio Code [管理者]
core.py
135 :obj: ndarray : フィルター部と影のマスク (グレースケール画像)
136 :obj: ndarray : フィルター部の影の上端部 (グレースケール画像)
137 :obj: ndarray : フィルター部の影 (グレースケール画像)
138 ...
139 shd_length = int(np.round(shd_length / scl))
140 shd_ext = shd_length / 16
141 src_height, src_width = src.shape[:2]
142 cnt, hrc = cv2.findContours(src, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
143 img_shd = copy.copy(src)
144 cv2.drawContours(img_shd, cnt, -1, (255, 255, 255), int(shd_ext))
145 img_shd_fst = copy.copy(img_shd)
146 img_shd = np.insert(img_shd, src_height-1, 0, axis=0)
147 img_shd = np.delete(img_shd, 0, 0)
148 img_shd = cv2.subtract(img_shd_fst, img_shd)
149 shd_pos = np.where(img_shd == 255)
150 shd_i = np.array([0]).astype('uint8')
151 for i, j in zip(shd_pos[0], shd_pos[1]):
152     shd_i = np.append(shd_i, i*src_width + j)
153 shd_coef = ((1/2)**(1/(shd_length/8)))
154 img_shd = np.zeros((src_height+shd_length, src_width))
155 for i in reversed(range(shd_length)):
156     shd_color = int(np.round(256 * shd_coef**(i+1)))
157     np.put(img_shd, shd_i + src_width*(i+1), shd_color)
158 img_shd = np.delete(img_shd, slice(src_height, src_height+shd_length))
159 img_shd = img_shd.astype(np.uint8)
160 imgflt = cv2.add(img_shd_fst, img_shd)
161 return imgflt, img_shd_fst, img_shd
162
163
```



## 過去の検査データ

検査前フォルダ

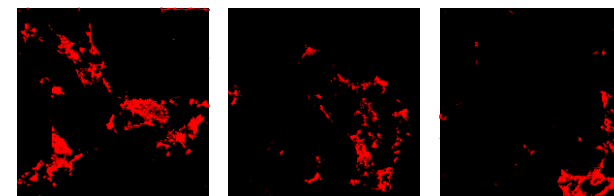


001.jpg

002.jpg

003.jpg

検査後フォルダ



001.png

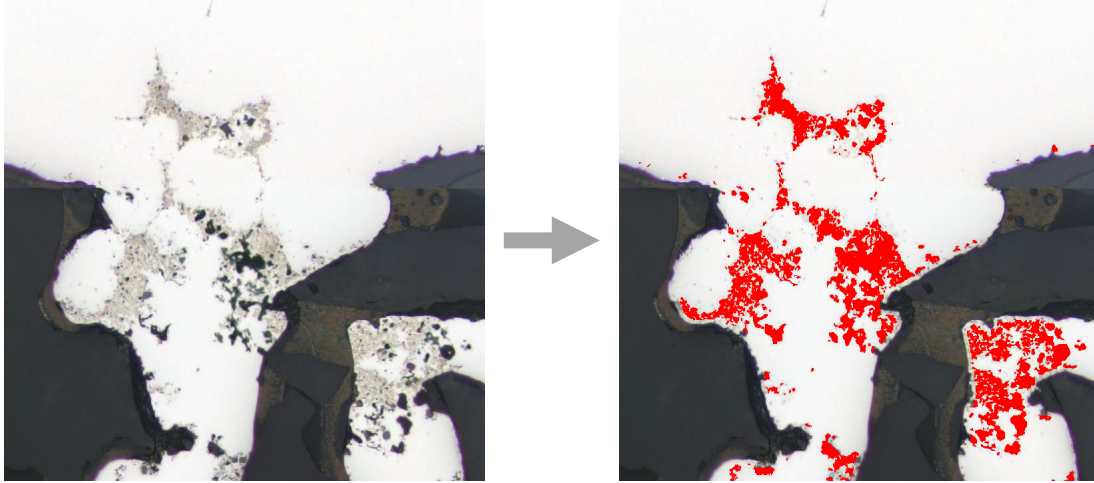
002.png

003.png

もしかして機械学習に  
使えるのでは？



実現したいこと



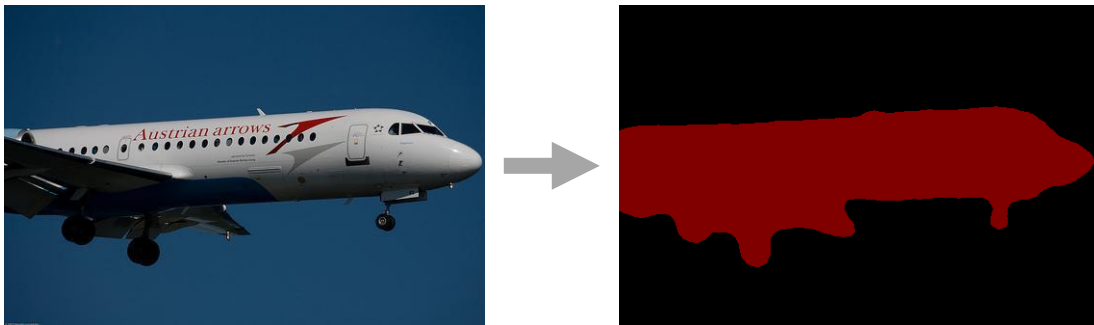
データはあるけど...  
どんな機械学習をすれば  
良いのだろうか



Amazon SageMaker

機械学習もセマンティック  
セグメンテーションも  
簡単にできますよ

セマンティックセグメンテーション [1]

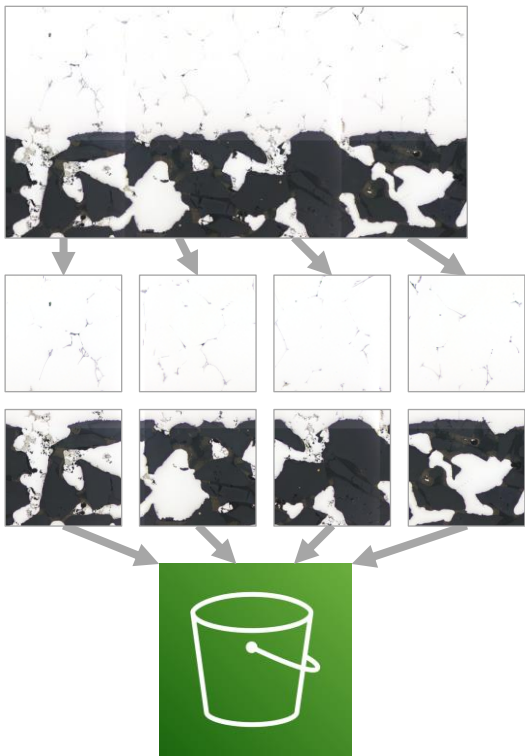
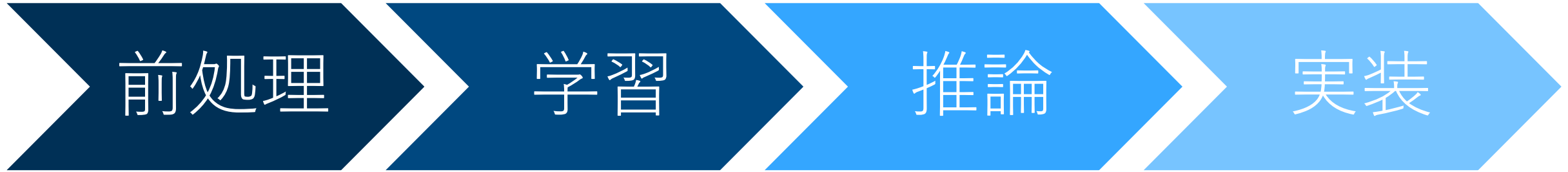


これだ！

**SageMaker** を使って  
画像検査自動化に挑戦！



[1] "1. Getting Started with FCN Pre-trained Models — gluoncv 0.11.0 documentation".  
[https://cv.gluon.ai/build/examples\\_segmentation/demo\\_fcn.html#sphx-glr-build-examples-segmentation-demo-fcn-py](https://cv.gluon.ai/build/examples_segmentation/demo_fcn.html#sphx-glr-build-examples-segmentation-demo-fcn-py), (accessed 2023/08/21).



Amazon S3



Amazon SageMaker

### ハイパーパラメータ

ハイパーパラメータを使用してトレーニングを細かく制御できます。選択したアルゴリズムのデフォルトパラメータ範囲が設定されました。 [詳細はこちら](#)

キー	値
backbone	resnet-50
use_pretrained_model	True
algorithm	fcn
lr_scheduler	poly
crop_size	240
num_classes	2

トレーニングジョブの作成

### バッチ変換ジョブの設定

ジョブ名  
Sample-Job  
最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

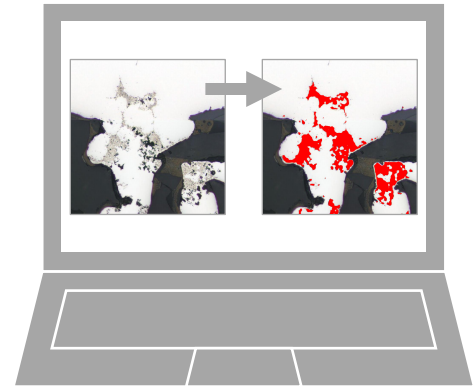
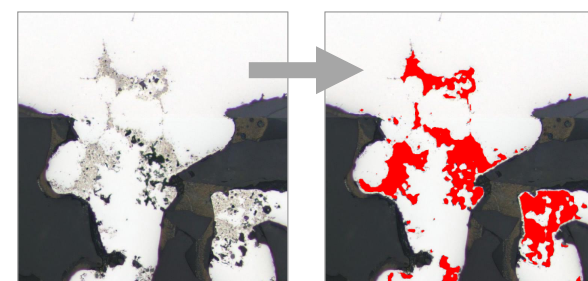
モデル名  
Sample-Model   
最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

インスタンスタイプ  
mlc5.xlarge

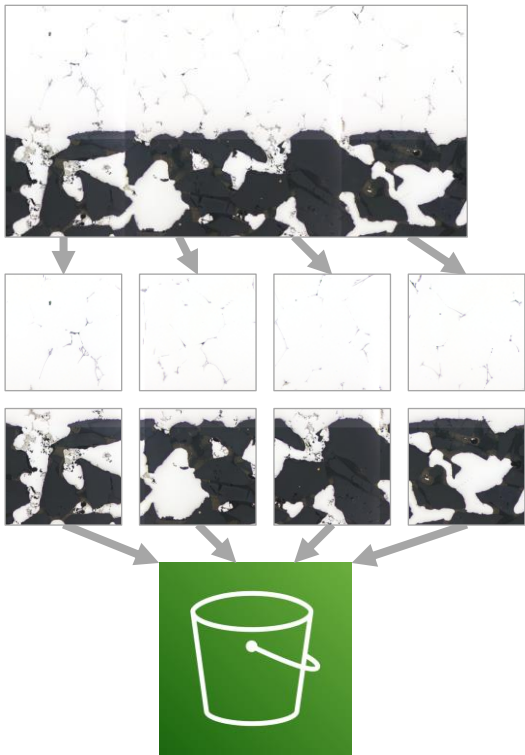
インスタンス数  
1

暗号化キー - オプション  
データを暗号化します。既存の KMS キーを選択するか、キーの ARN を入力します。  
カスタム暗号化なし

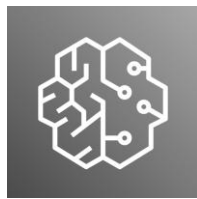
### ジョブの作成







Amazon S3



Amazon SageMaker

### ハイパーパラメータ

ハイパーパラメータを使用してトレーニングを細かく制御できます。選択したアルゴリズムのデフォルトパラメータ範囲が設定されました。詳細はこちら [🔗](#)

キー	値
backbone	resnet-50
use_pretrained_model	True
algorithm	fcn
lr_scheduler	poly
crop_size	240
num_classes	2

トレーニングジョブの作成

### バッチ変換ジョブの設定

ジョブ名  
Sample-Job

最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

モデル名  
Sample-Model

最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

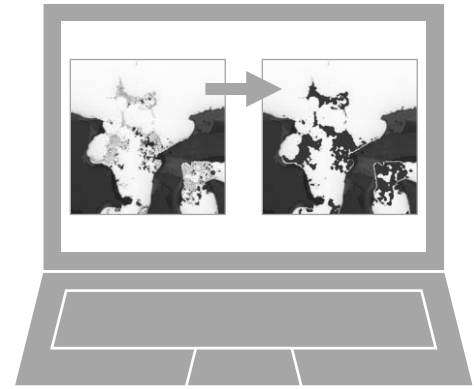
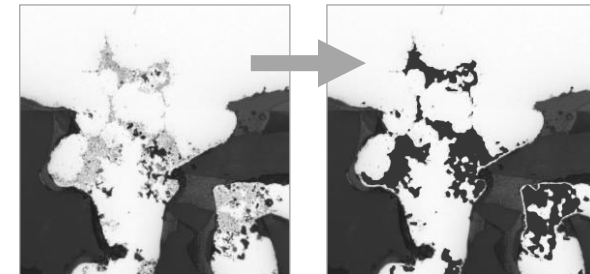
インスタンスタイプ  
mlc5.xlarge

インスタンス数  
1

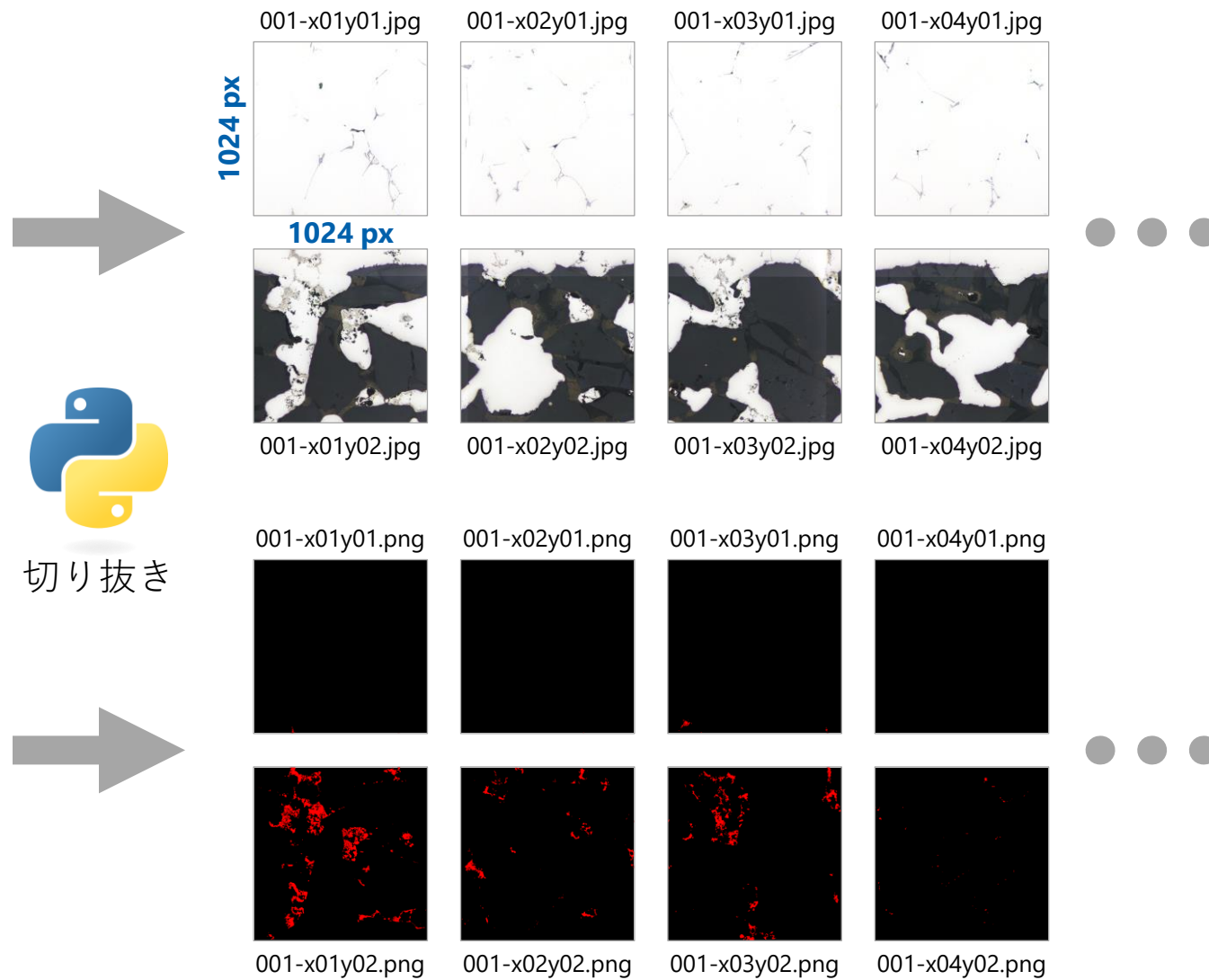
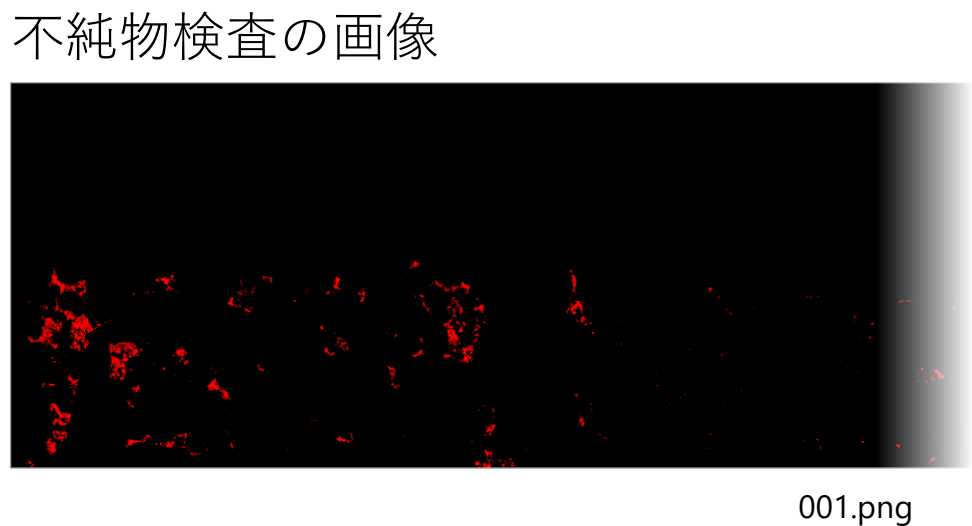
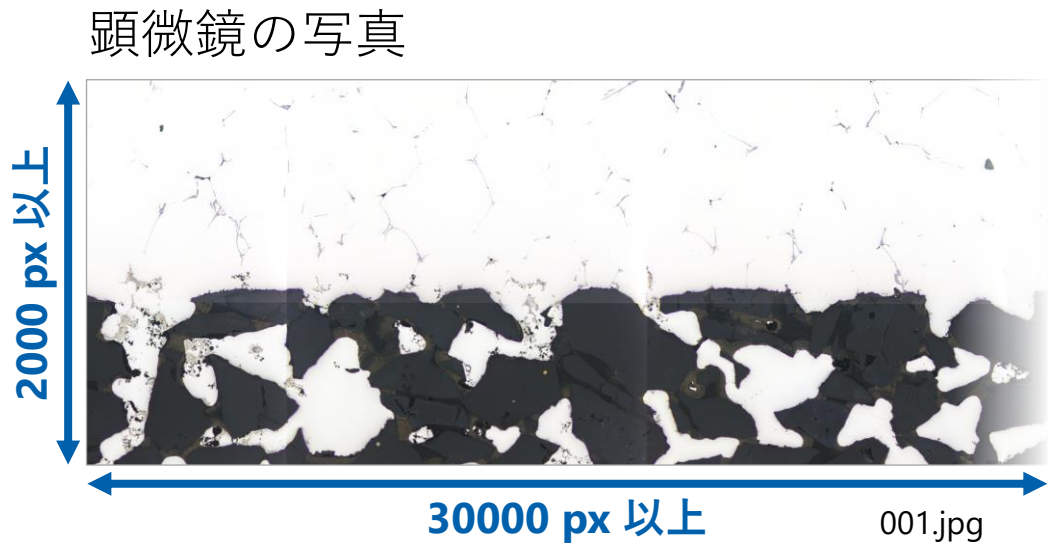
暗号化キー - オプション  
データを暗号化します。既存の KMS キーを選択するか、キーの ARN を入力します。

カスタム暗号化なし

### ジョブの作成

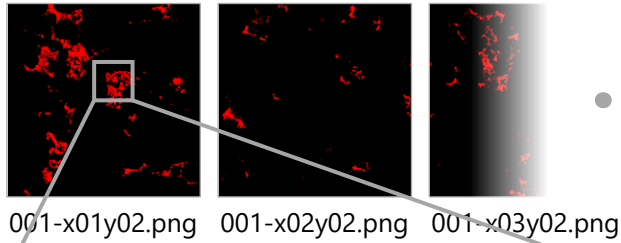


# 前処理 - 学習できる画像サイズに切り抜く



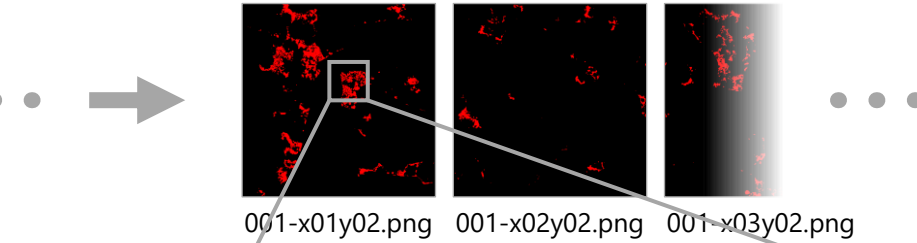
# 前処理 - PNG をインデックスカラーに変換する

不純物検査の画像



#000000	#000000	#000000	#000000
#000000	#FF0000	#FF0000	#FF0000
#000000	#FF0000	#FF0000	#000000
#000000	#FF0000	#FF0000	#000000

RGB



0	0	0	0
0	1	1	1
0	1	1	0
0	1	1	0

インデックスカラー



[1]

1	1	1	0	0	0
1	1	1	1	0	0
1	1	1	1	1	0
1	1	1	1	1	0
0	1	1	1	0	0
0	0	0	0	0	0

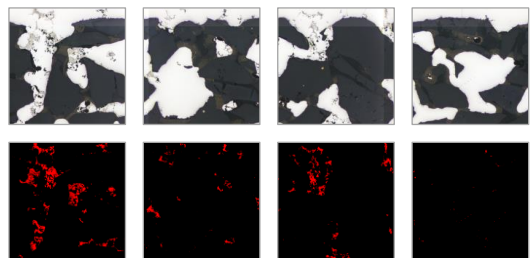
飛行機: 1
背景: 0

セマンティックセグメンテーションではピクセルごとに分類を行う [2]

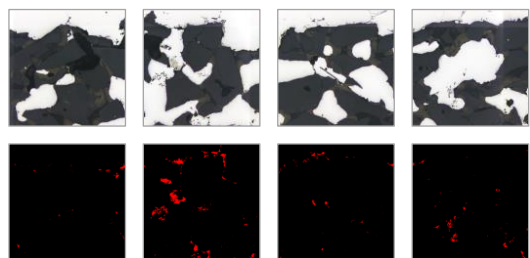
[1] "1. Getting Started with FCN Pre-trained Models — gluoncv 0.11.0 documentation". [https://cv.gluon.ai/build/examples\\_segmentation/demo\\_fcn.html#sphx-glr-build-examples-segmentation-demo-fcn-py](https://cv.gluon.ai/build/examples_segmentation/demo_fcn.html#sphx-glr-build-examples-segmentation-demo-fcn-py), (accessed 2023/08/21).

[2] Jeremy Jordan. "An overview of semantic image segmentation.". 2018/05/21. <https://www.jeremyjordan.me/semantic-segmentation/>, (accessed 2023/08/21).

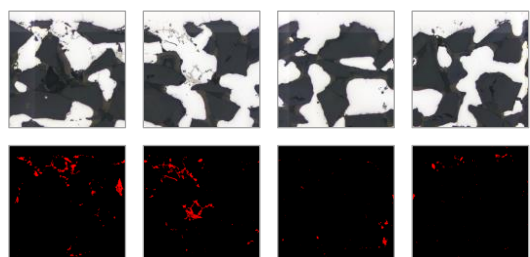
# 前処理 - フォルダ分け, S3 にアップロード



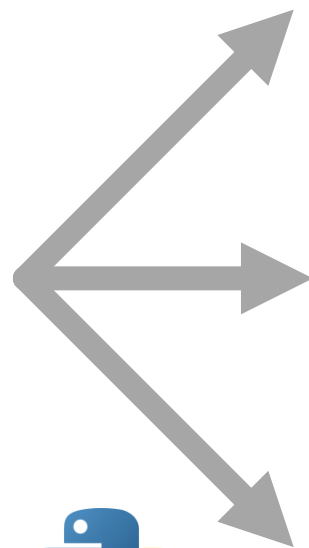
001.jpg, 001.png



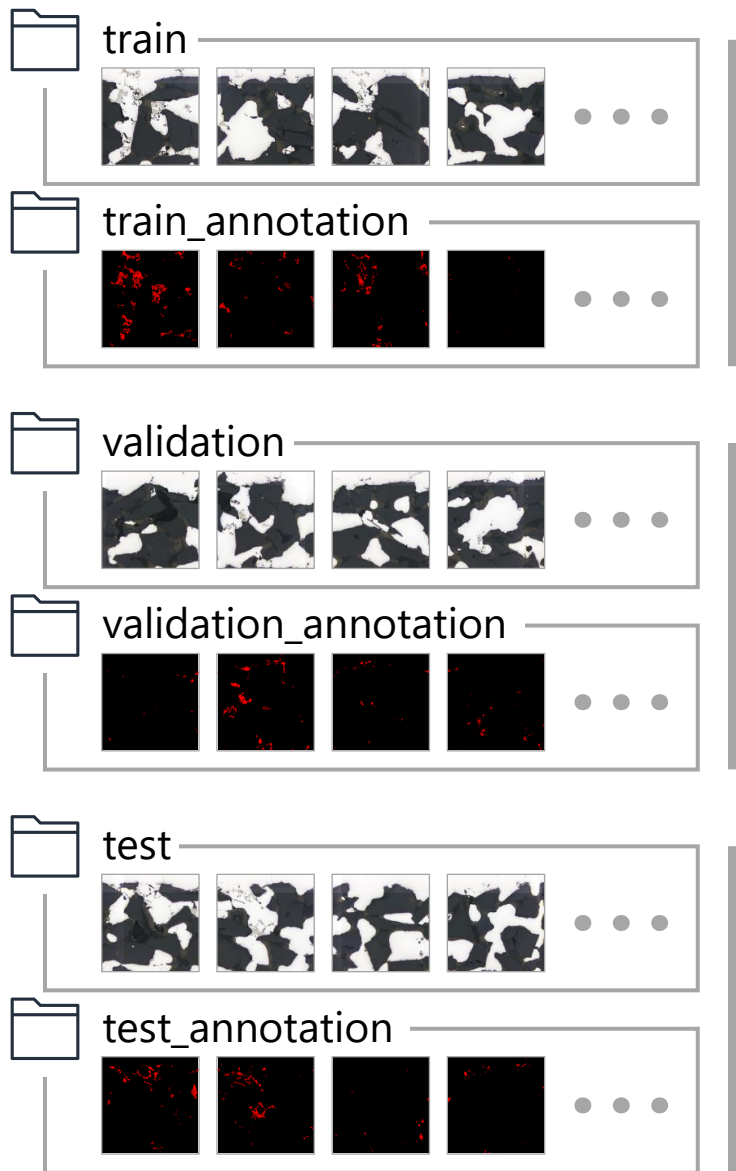
002.jpg, 002.png



003.jpg, 003.png



無作為に  
フォルダ分け



ファイル数

16

対

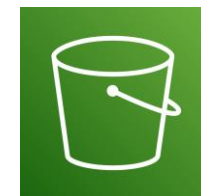
4

対

5



アップロード

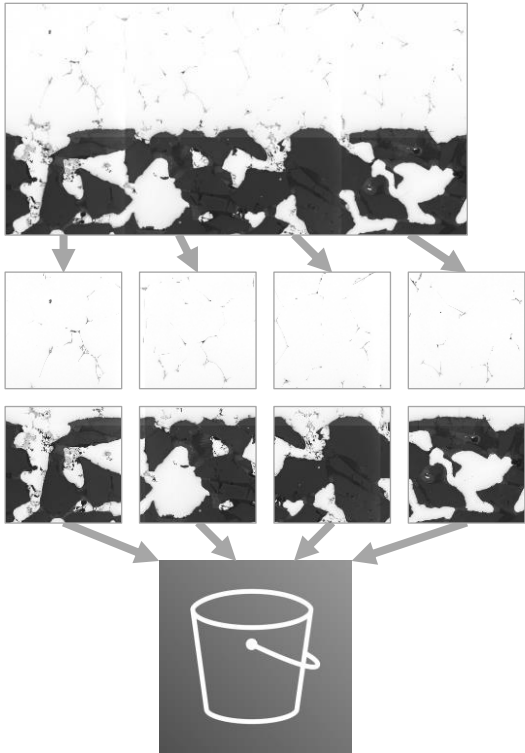


Amazon S3

前処理完了







Amazon S3



Amazon SageMaker

### ハイパーパラメータ

ハイパーパラメータを使用してトレーニングを細かく制御できます。選択したアルゴリズムのデフォルトパラメータ範囲が設定されました。 [詳細はこちら](#)

キー	値
backbone	resnet-50
use_pretrained_model	True
algorithm	fcn
lr_scheduler	poly
crop_size	240
num_classes	2

トレーニングジョブの作成

### バッチ変換ジョブの設定

ジョブ名  
Sample-Job  
最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

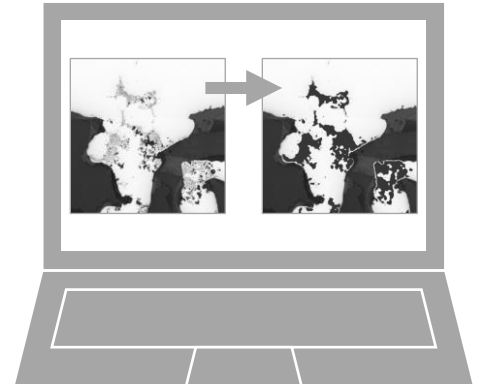
モデル名  
Sample-Model   
最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

インスタンスタイプ  
mlc5.xlarge

インスタンス数  
1

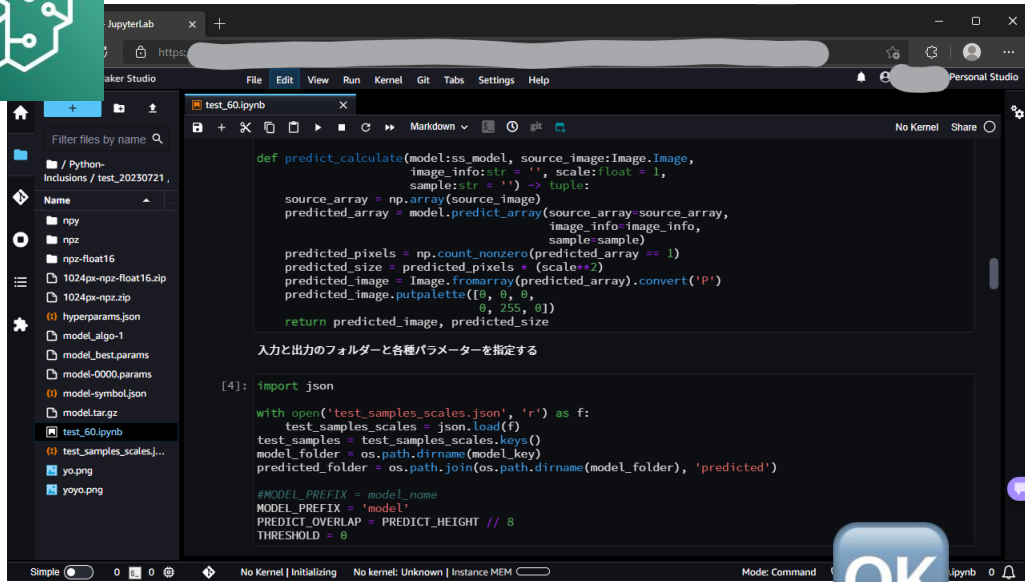
暗号化キー - オプション  
データを暗号化します。既存の KMS キーを選択するか、キーの ARN を入力します。  
カスタム暗号化なし

**ジョブの作成**

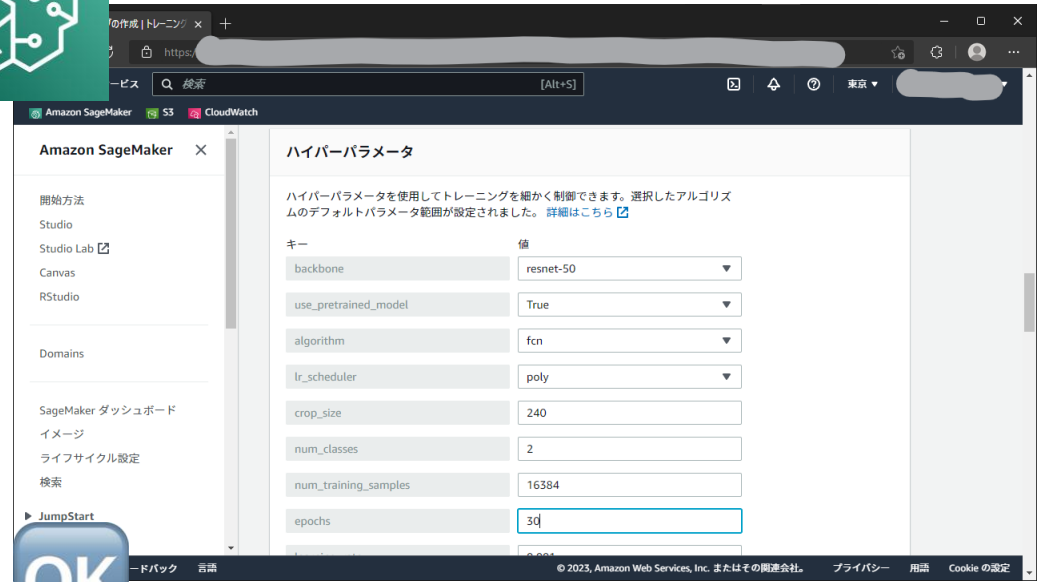




## SageMaker Studio



## SageMaker コンソール

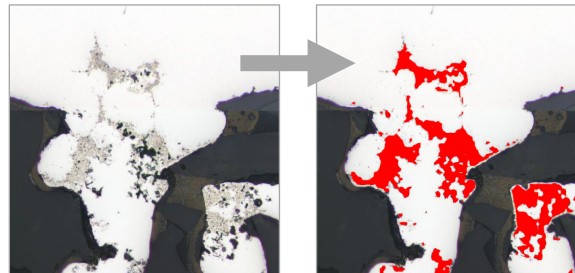


OK

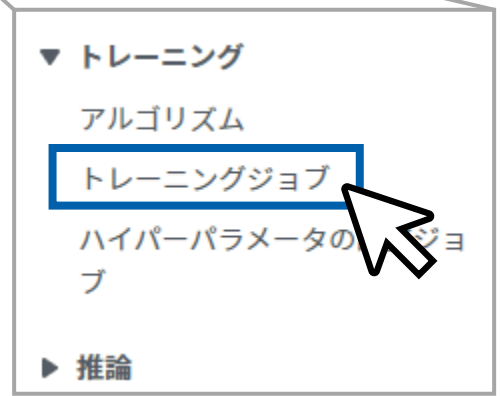
OK

どちらでも

セマンティックセグメンテーションできる！



本日はこちら



トレーニングジョブの作成



設定項目が多い！



- 👉 ジョブ設定
  - ネットワーク
- 👉 ハイパーパラメータ
- 👉 入力データ設定
  - チェックポイント設定
  - 出力データ設定
  - マネージド型スポットのトレーニング
  - タグ

でも実は...

既に決まっている値を入力・選択するだけ



### ジョブ設定

ジョブ名

好きな名前

最大 63 文字の英数字を使用できます。ハイフン (-) は含めることができますが、スペースは含めないでください。一つのAWS リージョンのアカウント内で一意である必要があります。

IAM ロール

Amazon SageMaker では、ユーザーに代わって他のサービスを呼び出すアクセス許可が必要です。ロールを選択するか、AmazonSageMakerFullAccess AWS に IAM ポリシーがアタッチされたロールを作成させます。

ロール選択

ロール作成ウィザードを使用してロールを作成

### リソース設定

インスタンスタイプ

ml.p3.2xlarge を選択 [3]

インスタンス数

インスタンスあたりの追加のストレージボリューム (GB)

インスタンス数: 1  
ボリューム: 50 GB

キーブアライブ期間

使用 SageMaker トレーニングマネージドウォームプール

seconds

最大ランタイムは 1 時間 (60 分または 3,600 秒) です。

暗号化キー - オプション

データを暗号化します。既存の KMS キーを選択するか、キーの ARN を入力します。

カスタム暗号化なし

### 停止条件

最大ランタイム

seconds

### アルゴリズムのオプション

Amazon SageMaker 組み込みアルゴリズム、独自のアルゴリズム、または AWS Marketplace のサードパーティアルゴリズムを使用してください。

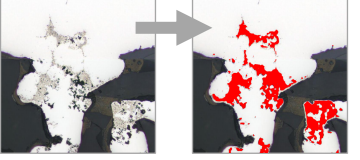
アルゴリズムのソース

- Amazon SageMaker 組み込みアルゴリズム [詳細はこちら](#)
- 独自のアルゴリズムリソース
- ECRの独自のアルゴリズムコンテナ [詳細はこちら](#)
- AWS Marketplace からのサブスクリプションアルゴリズム

アルゴリズムの選択

- Vision - Semantic Segmentation (MxNet)
- Tabular - XGBoost : v1.3
- Tabular - Linear Learner
- Tabular - K Nearest Neighbors (KNN)
- Tabular - Factorization Machines
- Tabular - Object2Vec
- Vision - Image Classification (MxNet)
- Vision - Object Detection (MxNet)
- Vision - Semantic Segmentation (MxNet)**
- Clustering - K-Means
- Time Series Forecast - DeepAR
- Text Classification & Text Embedding - Blazing Text
- Text Transformation - Sequence to Sequence (MxNet)
- Text Topic Modeling - Neural Topic Modeling (NTM)
- Text Topic Modeling - Latent Dirichlet Allocation (LDA)
- Dimensionality Reduction - Principal Component Analysis (PCA)
- Anomaly Detection - Random Cut Forest
- Anomaly Detection - IP Insights

セマンティックセグメンテーション



[3] "Amazon SageMaker Semantic Segmentation Algorithm — Amazon SageMaker Examples 1.0.0 documentation". [https://sagemaker-examples.readthedocs.io/en/latest/introduction\\_to\\_amazon\\_algorithms/semantic\\_segmentation\\_on\\_pascalvoc/semantic\\_segmentation\\_pascalvoc.html](https://sagemaker-examples.readthedocs.io/en/latest/introduction_to_amazon_algorithms/semantic_segmentation_on_pascalvoc/semantic_segmentation_pascalvoc.html), (accessed 2023/08/21).



初めは初期設定のままで OK  
さらに精度が必要になったら調整する



今回見つけたベストなハイパーパラメータ



### ハイパーパラメータ

ハイパーパラメータを使用してトレーニングを細かく制御できます。選択したアルゴリズムのデフォルトパラメータ範囲が設定されました。 [詳細はこちら](#)

キー	値
backbone	resnet-101
use_pretrained_model	False
algorithm	fcn
lr_scheduler	poly
crop_size	256
num_classes	2
num_training_samples	16384
epochs	30
learning_rate	0.0001

• 不純物  
• 不純物でない  
の 2 種類に分けたいので  
num\_classes に 2 と入力

train フォルダにある  
ファイル数またはそれ以下であれば OK

gamma1	
optimizer	rmsprop
weight_decay	
momentum	
mini_batch_size	16
validation_mini_batch_size	16
early_stopping_min_epochs	10
early_stopping_patience	2
early_stopping_tolerance	
early_stopping	True

optimizer に rmsprop を  
選んだので空欄で OK

train

train\_annotation

validation

validation\_annotation

File

image/png

チャンネルの追加

貼り付け

Amazon S3 > バケット > [redacted]

At-Inclusions-SS-Test/

オブジェクト | プロパティ

オブジェクト (6)

S3 URI をコピー

URL をコピー

ダウンロード

名前	タイプ	最終更新日時
test_annotation/	フォルダ	-
test/	フォルダ	-
train_annotation/	フォルダ	-
<input checked="" type="checkbox"/> train/	フォルダ	-
validation_annotation/	フォルダ	-
validation/	フォルダ	-

# 学習 - トレーニングを開始する

**チェックポイント設定 - オプション**  
アルゴリズムは、チェックポイントを定期的に生成します。チェックポイントはこの場所に保存され、管理されたスポットトレーニングジョブを再開するために使用されます。 [詳細はこちら](#)。

S3 の場所  
s3://[redacted]/checkpoint  
パスを見つけるには、 [Amazon S3 に移動](#)

**出力データ設定**

S3 出力パス  
s3://[redacted]/output

**マネージド型スポットのトレーニング**      **学習 1 回 (8 時間)**  
**4300 円 → 1300 円**

マネージド型スポットトレーニングの有効化 - オプション  
開始と終了時間に柔軟性があるジョブのコンピューティングコストをセーブします。Amazon SageMaker は、このジョブを実行するためにのみ、予備の容量を使用します。 [詳細はこちら](#)

トレーニングジョブの作成

トレーニングジョブ 情報        アクション ▼   

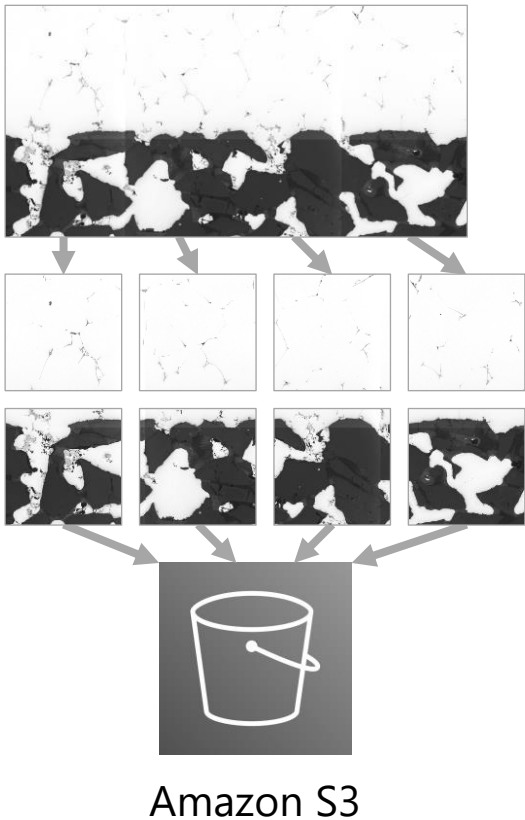
検索トレーニングジョブ    < 1 >    ⚙️

名前 ▼	作成時刻 ▼	期間	ジョブのステータス ▼	ウォームアップ
SS-AL-Inclusions-155samples-deeplab	2023/8/7 8:50:09	-	InProgress	-



名前 ▲	タイプ ▼	最終更新日時 ▼	サイズ ▼
model.tar.gz	gz	2023/07/21 07:17:06 AM JST	385.6 MB

モデルが保存される



### Amazon SageMaker

**ハイパーパラメータ**

ハイパーパラメータを使用してトレーニングを細かく制御できます。選択したアルゴリズムのデフォルトパラメータ範囲が設定されました。詳細はこちら [🔗](#)

キー	値
backbone	resnet-50
use_pretrained_model	True
algorithm	fcn
lr_scheduler	poly
crop_size	240
num_classes	2

**トレーニングジョブの作成**

### バッチ変換ジョブの設定

ジョブ名  
Sample-Job  
最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

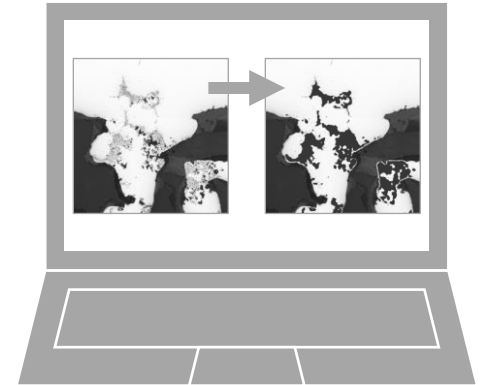
モデル名  
Sample-Model   
最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

インスタンスタイプ  
mlc5.xlarge

インスタンス数  
1

暗号化キー - オプション  
データを暗号化します。既存の KMS キーを選択するか、キーの ARN を入力します。  
カスタム暗号化なし

**ジョブの作成**







S3 に保存されたモデルを  
SageMaker で使えるようにする

トレーニングジョブ 情報

検索トレーニングジョブ   1

名前	作成時刻	期間	ジョブのステータス	ウォームプール
SS-AI- Inclusions- 155samples- rmsprop	2023/7/18 15:54:39	a day	✔ Completed	-

モデルの作成

Amazon SageMaker > モデル > モデルの作成

## モデルの作成

Amazon SageMaker にモデルをデプロイするには、最初にモデルアーティファクトの場所と推論コードを指定してモデルを作成します。参照[Amazon SageMaker ホスティングサービスでモデルをデプロイする](#) [API の詳細](#)

### モデル設定

モデル名  **好きな名前**

最大 63 文字の英数字を使用できます。ハイフン (-) は含めることができますが、スペースは含めないでください。一つのAWS リージョンのアカウント内で一意である必要があります。

IAM ロール  
Amazon SageMaker では、ユーザーに代わって他のサービス呼び出すアクセス許可が必要です。ロールを選択するか、[AmazonSageMakerFullAccess](#) AWS に IAM ポリシーがアタッチされたロールを作成させます。

他はそのまま OK

モデルの作成

成功! モデルを作成しました。  
このモデルを使用するには、[Create endpoint] を選択します。

Amazon SageMaker > モデル

モデル

検索モデル   1

名前	ARN	作成時刻
Sample-Model-0001	arn:aws:sagemaker:ap-northeast-1: [redacted]:model/sample-model-0001	2023/8/7 11:26:32



モデルの精度を評価したい  
そのために大量の画像を一気に予測させる

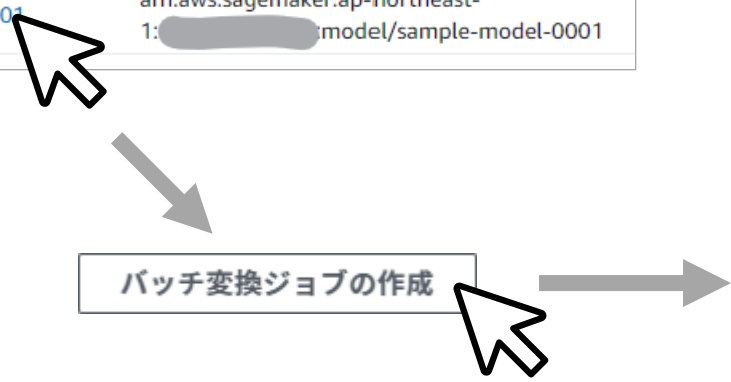
成功! モデルを作成しました。  
このモデルを使用するには、[Create endpoint] を選択します。

Amazon SageMaker > モデル

モデル

検索モデル

名前	ARN
Sample-Model-0001	arn:aws:sagemaker:ap-northeast-1: [redacted] model/sample-model-0001



Amazon SageMaker > バッチ変換ジョブ > バッチ変換ジョブの作成

## バッチ変換ジョブの作成

変換ジョブは、モデルを使用してデータを変換し、指定された場所に結果を保存します。 [詳細はこちら](#)

### バッチ変換ジョブの設定

ジョブ名  **好きな名前**

最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

モデル名

最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

インスタンスタイプ  **ml.c5.xlarge を選択 [3]**

インスタンス数  **インスタンス数: 1**

暗号化キー - オプション  
データを暗号化します。既存の KMS キーを選択するか、キーの ARN を入力します。

▶ 追加設定

[3] "Amazon SageMaker Semantic Segmentation Algorithm — Amazon SageMaker Examples 1.0.0 documentation". [https://sagemaker-examples.readthedocs.io/en/latest/introduction\\_to\\_amazon\\_algorithms/semantic\\_segmentation\\_pascalvoc/semantic\\_segmentation\\_pascalvoc.html](https://sagemaker-examples.readthedocs.io/en/latest/introduction_to_amazon_algorithms/semantic_segmentation_pascalvoc/semantic_segmentation_pascalvoc.html), (accessed 2023/08/21).

# 推論 - バッチ変換ジョブを開始する

**入力データ設定**

S3 データタイプ: S3Prefix  
分割タイプ: None

圧縮: None

コンテンツタイプ オプション: image/jpeg

S3 の場所: s3://[redacted]/test

**出力データ設定**

S3 出力パス: s3://[redacted]/predicted

組み合わせ: None

追加設定

許可 - オプション: image/png

暗号化キー - オプション: カスタム暗号化なし

S3 URI をコピー

名前	タイプ
test_annotation/	フォルダ
<input checked="" type="checkbox"/> test/	フォルダ
train_annotation/	フォルダ
train/	フォルダ
validation_annotation/	フォルダ
validation/	フォルダ

ジョブの作成

バッチ変換ジョブ Sample-Predict-0001 は正常に作成されました。

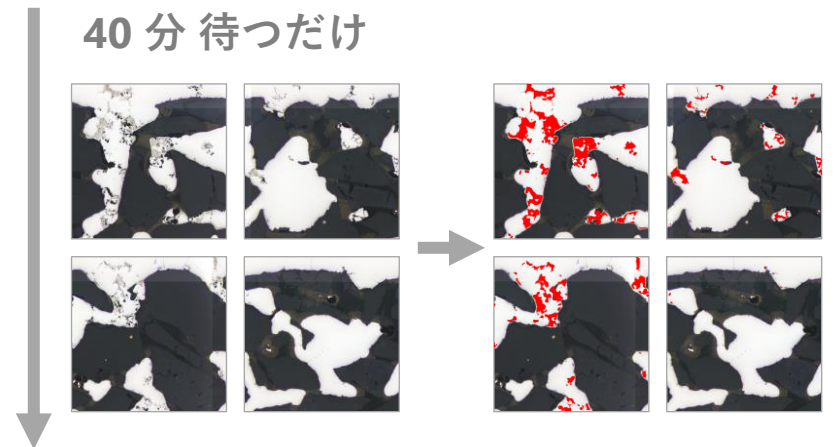
Amazon SageMaker > バッチ変換ジョブ

バッチ変換ジョブ [リフレッシュ] アクション [バッチ変換ジョブの作成]

検索バッチ変換ジョブ

名前	ステータス	期間	作成時刻
Sample-Predict-0001	InProgress	a few seconds	2023/8/7 15:42:12

InProgress 予測中



Completed 予測完了

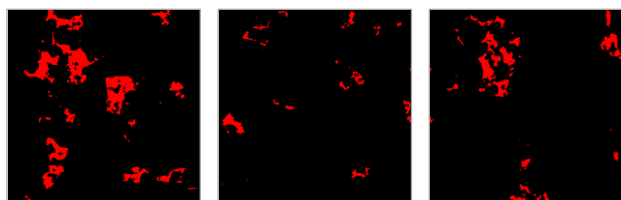
🗑️ 予測結果フォルダ

名前	タイプ
001-x01y01.jpg.out	out
001-x01y02.jpg.out	out
001-x01y03.jpg.out	out
001-x01y04.jpg.out	out
001-x02y01.jpg.out	out
001-x02y02.jpg.out	out
001-x02y03.jpg.out	out
001-x02y04.jpg.out	out

jpg.out って何のファイル...? 🤔

中身は .png (インデックスカラー) であり  
ファイル名が .jpg.out になっているだけ!

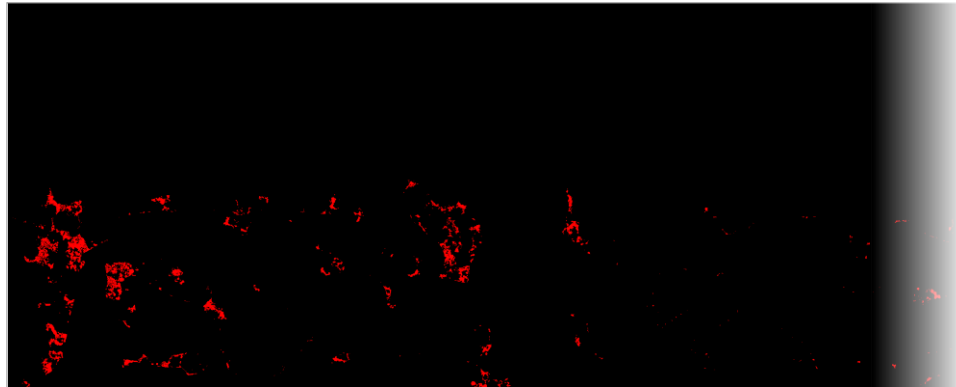
↓  
ファイル名変更



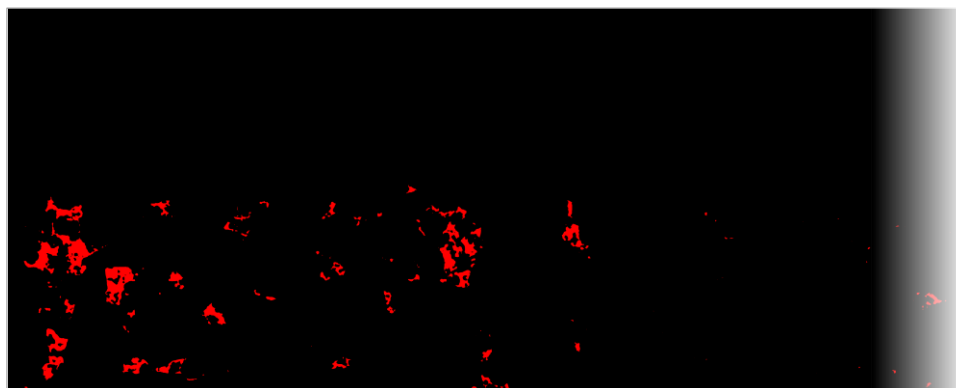
001-x01y02.png    001-x02y02.png    001-x03y02.png

画像連結

熟練者の検査結果



モデルの予測結果



\*1 誤差とは熟練者が算出した画像上の不純物面積に対する平均絶対誤差率のこと



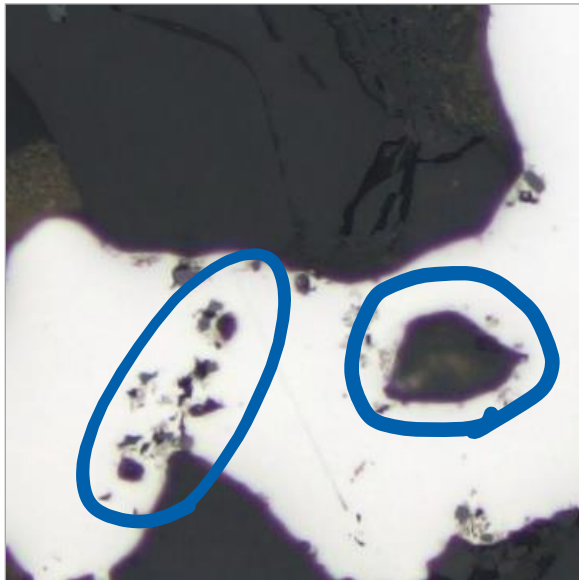
面積計算  
比較

最初のモデルは  
誤差 20% \*1

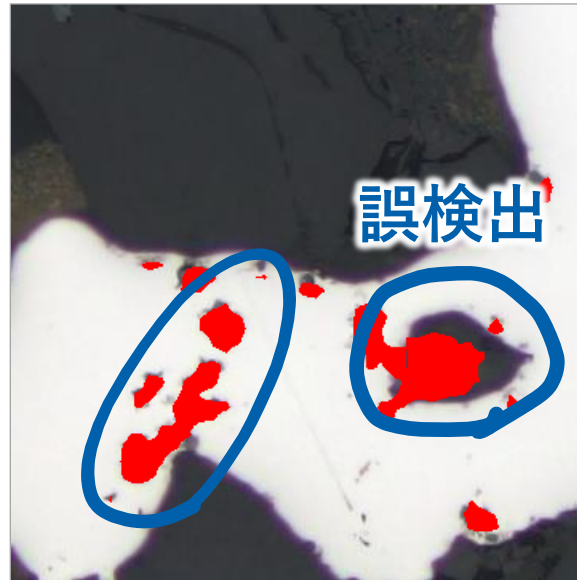




検査前



最初 の予測結果



誤差 20% \*1

調整後 の予測結果



誤差 8% \*1

backbone: resnet-50  
use\_pretrained\_model: True  
crop\_size: 240



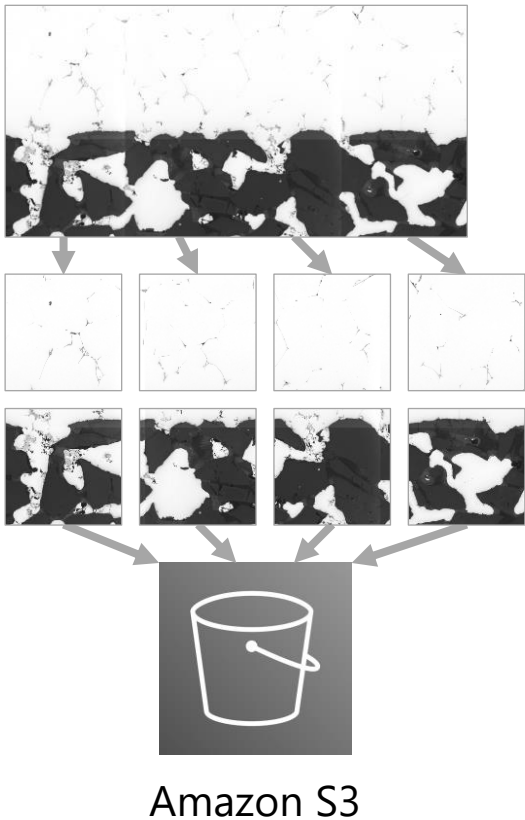
resnet-101  
False  
256

熟練者のチェック  
予測精度 OK



ハイパーパラメータを調整

\*1 誤差とは熟練者が算出した画像上の不純物面積に対する平均絶対誤差率のこと



## Amazon SageMaker

**ハイパーパラメータ**

ハイパーパラメータを使用してトレーニングを細かく制御できます。選択したアルゴリズムのデフォルトパラメータ範囲が設定されました。詳細はこちら [🔗](#)

キー	値
backbone	resnet-50
use_pretrained_model	True
algorithm	fcn
lr_scheduler	poly
crop_size	240
num_classes	2

**トレーニングジョブの作成**

### バッチ変換ジョブの設定

ジョブ名  
Sample-Job

最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

モデル名  
Sample-Model

最大 63 文字の英数字。ハイフン (-) は使用できますが、スペースを含めることはできません。同じ AWS リージョンのお客様のアカウント内で一意である必要があります。

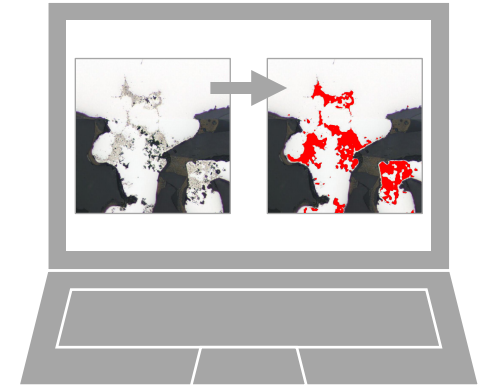
インスタンスタイプ  
mlc5.xlarge

インスタンス数  
1

暗号化キー - オプション  
データを暗号化します。既存の KMS キーを選択するか、キーの ARN を入力します。

カスタム暗号化なし

**ジョブの作成**



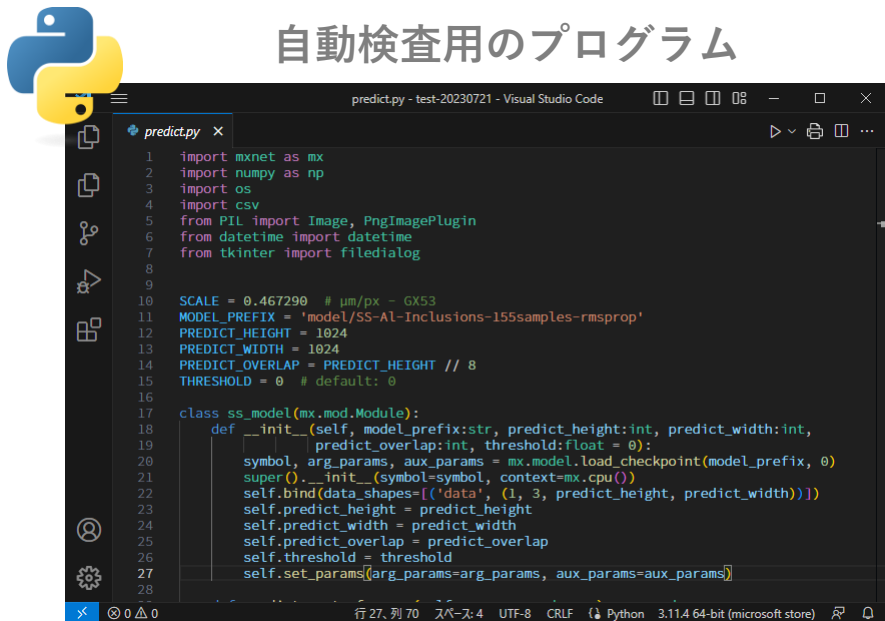


名前	タイプ	最終更新日時
model.tar.gz	gz	2023/07/21 07:17:06 AM JST

SageMaker で作成したモデル



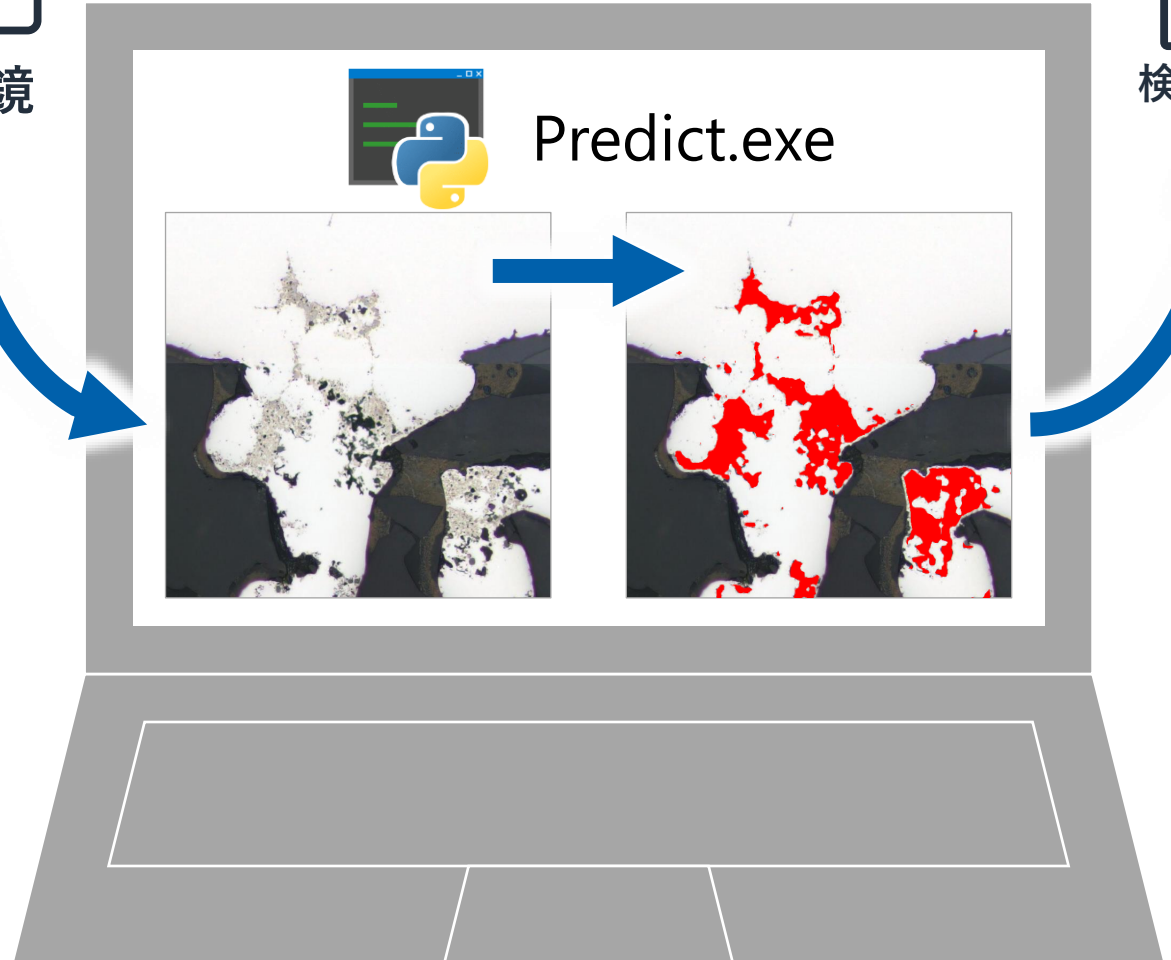
自動検査用のプログラム



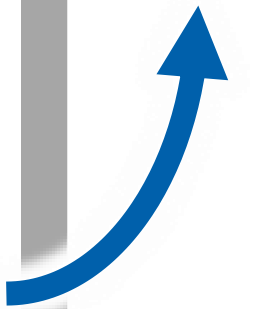
```
predict.py ×
1 import mxnet as mx
2 import numpy as np
3 import os
4 import csv
5 from PIL import Image, PngImagePlugin
6 from datetime import datetime
7 from tkinter import filedialog
8
9
10 SCALE = 0.467290 # μm/px - GX53
11 MODEL_PREFIX = 'model/SS-Al-Inclusions-155samples-rmsprop'
12 PREDICT_HEIGHT = 1024
13 PREDICT_WIDTH = 1024
14 PREDICT_OVERLAP = PREDICT_HEIGHT // 8
15 THRESHOLD = 0 # default: 0
16
17 class ss_model(mx.mod.Module):
18     def __init__(self, model_prefix:str, predict_height:int, predict_width:int,
19                 predict_overlap:int, threshold:float = 0):
20         symbol, arg_params, aux_params = mx.model.load_checkpoint(model_prefix, 0)
21         super().__init__(symbol=symbol, context=mx.cpu())
22         self.bind(data_shapes=[('data', (1, 3, predict_height, predict_width))])
23         self.predict_height = predict_height
24         self.predict_width = predict_width
25         self.predict_overlap = predict_overlap
26         self.threshold = threshold
27         self.set_params([arg_params=arg_params, aux_params=aux_params])
28
```



顕微鏡

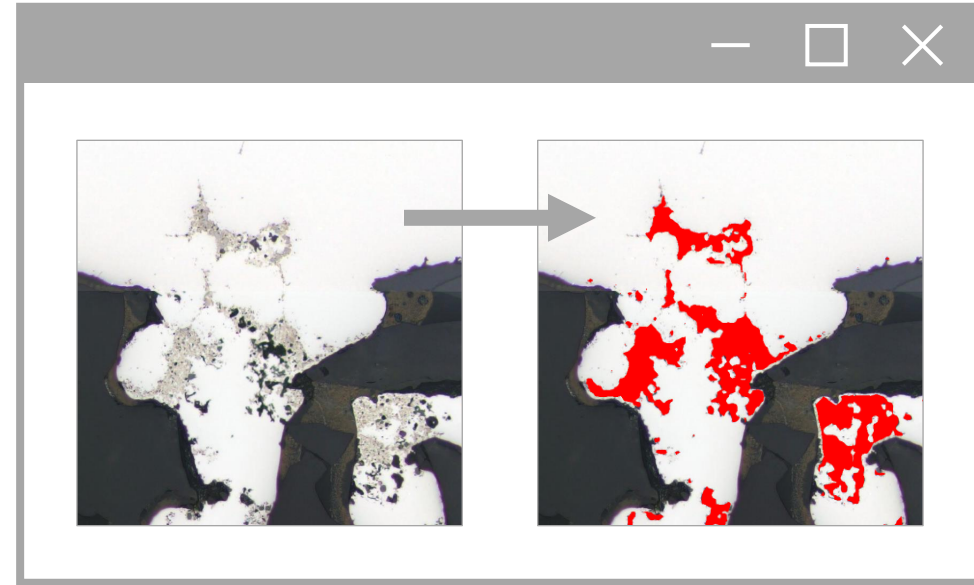
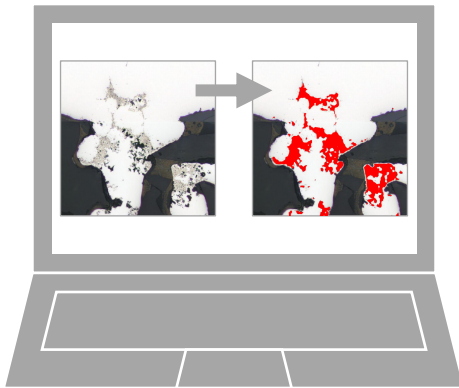


検査結果

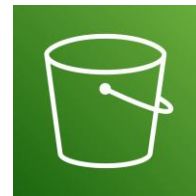


## モデルを随時アップデートできる 自動検査のオンラインサービス

オンプレミス



Amazon SageMaker



Amazon S3



AWS Lambda



AWS Step Functions



クラウドサービスの普及により、  
機械学習といった IT 専門家しか扱えなかった技術が、  
プログラミング初心者でも簡単に扱えるようになりました。

**実現したいものは実現したい人が作る**

AWS を活用すれば、自分達にとっての  
ベストクオリティを目指すことができます。