Google Cloud

JAX / Flax 入門

中井 悦司
Google Cloud, Solutions Architect

$ who am i

中井悦司 / Etsuji Nakai

Solutions Architect, Google Cloud

今回のセッションの背景

EvoJAX: あなたの課題を Neuroevolution の力で解く

2022年6月28日

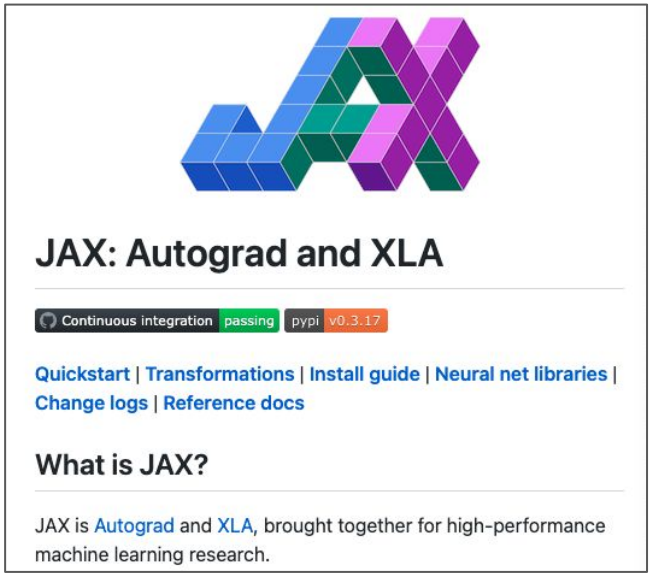
JAX はユーザーコードの簡略化や大規模な並列化・何桁もの高速化を可能にする、最近のGoogleで最も重要な機械学習 (ML) フレームワークの一つです。このフレームワークは、言語理解におけるPathways Language Model (PaLM) 、物理学・分子動力学シミュレーションにおける Brax や JAX MD などをはじめとして、近年に最先端(State-of-the-Art) の成果を示した研究でも利用されています。

JAX の人気が高まっていることは、今後様々な分野において JAX を利用した研究やツールが登場することに繋がります。この記事では、JAX 上で構築された EvoJAX について紹介します。これは、ハードウェア・アクセラレーション技術により高速化されたNeuroevolution のツールキットであり、微分可能でない要素を含むような複雑な問題の解決に利用できます。下図では、ユーザーが EvoJAX によって解くことができるタスクの例が示されています。(各タスクの詳細については、「応用例」のセクションを見てください。)

https://cloud.google.com/blog/ja/topics/developers-practitioners/evojax-bringing-power-neuroevolution-solve-your-problems

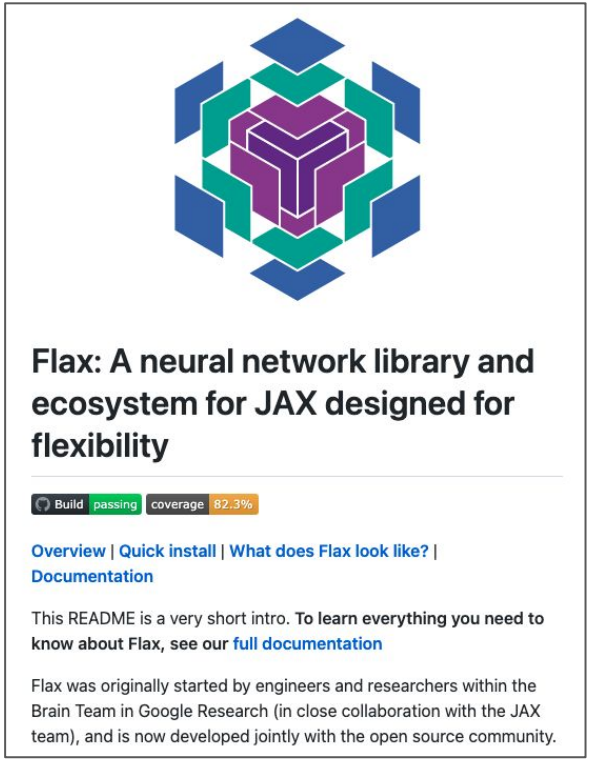
JAX とは？

- Google の AI 研究チーム（Google Brain）が開発した数値計算ライブラリー
- 機械学習のベースとなる計算処理を GPU / TPU で高速に実行可能
- NumPy とほぼ同じ関数（メソッド）を用意しており、NumPy を知っていればすぐに使える
- JAX 独自の機能の例
  - JIT コンパイラ: Python で定義した関数を GPU / TPU での計算に最適化されたバイナリーに事前コンパイル
  - 自動微分 : Python で定義した関数の微分（勾配ベクトル）を計算

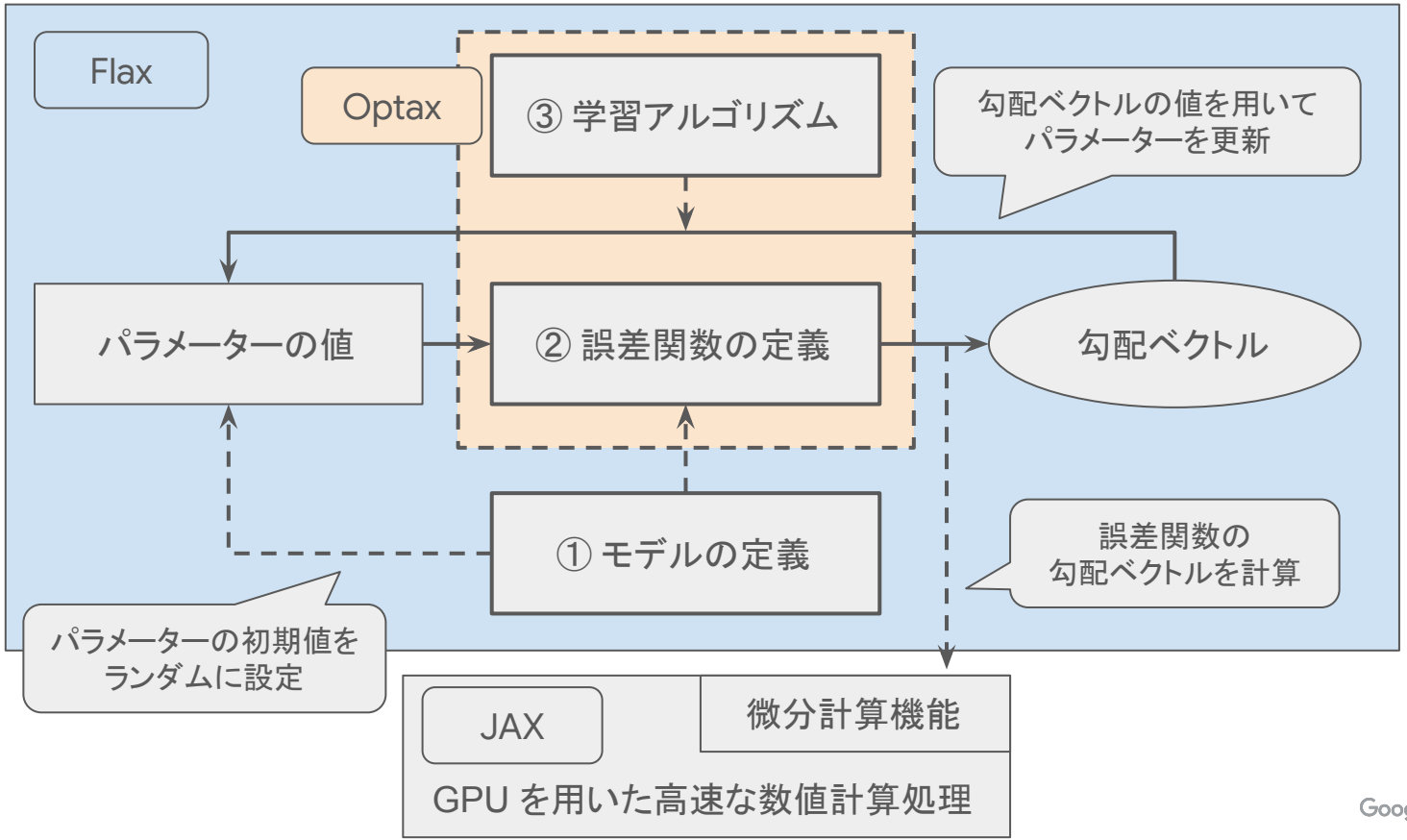


JAX は上位ライブラリーと組み合わせて使用

- JAX はあくまでも数値計算ライブラリーなので、それだけでは機械学習の処理（ニューラルネットワークの定義、勾配降下法による学習処理など）はできない
- さまざまな上位ライブラリーと組み合わせて使用する
  - Flax：ニューラルネットワークの構築と学習プロセスの管理機能を提供
  - Optax：誤差関数や勾配降下法のアルゴリズムをモジュールとして提供
  - などなど
- 今回は、JAX / Flax / Optax の組み合わせ例を紹介

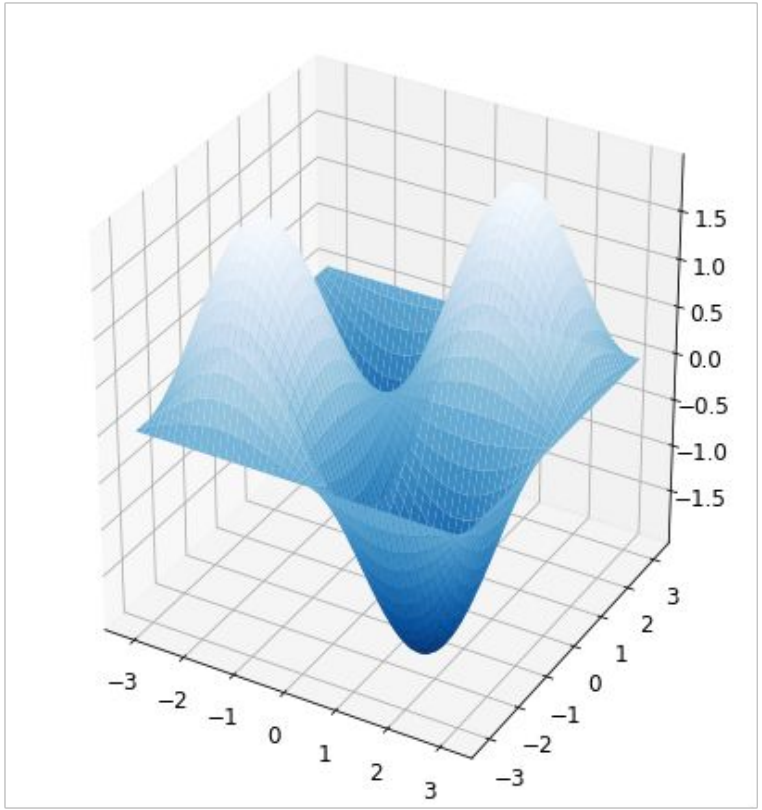


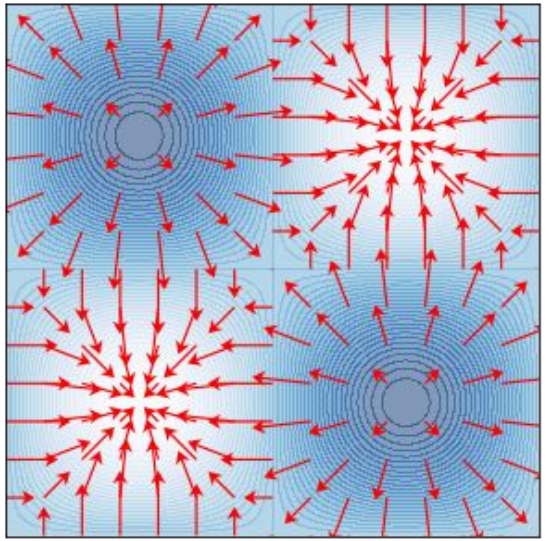
JAX / Flax / Optax の役割分担



JAX 入門

JAX で勾配ベクトルを計算！





https://gist.github.com/enakai00/739263fe6d3531fa0b7b18a7f3324892

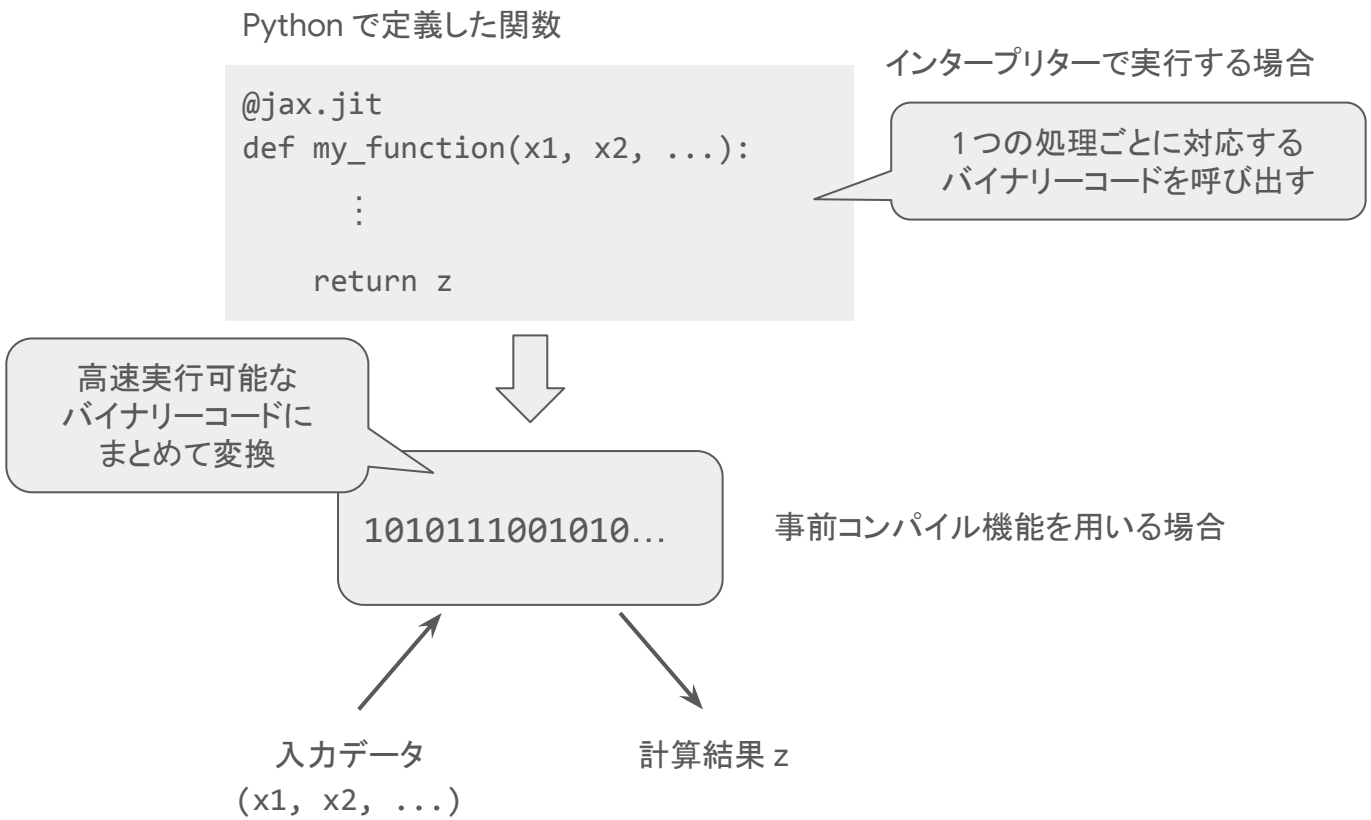
普通の Python の関数を微分可能

```
import jax
from jax import numpy as jnp

@jax.jit
def h(x1, x2):
    z = 2 * jnp.sin(x1) * jnp.sin(x2)
    return z

nabla_h = jax.jit(jax.grad(h, (0, 1)))

plot3d((-jnp.pi, jnp.pi), (-jnp.pi, jnp.pi), h, nabla_h)
```

2変数関数 $h(x_1, x_2) = 2 \times \sin x_1 \times \sin x_2$
を Python の関数として定義

勾配ベクトル $\left(\frac{\partial h}{\partial x_1}, \frac{\partial h}{\partial x_2}\right)$ を計算

JIT コンパイラによる事前コンパイル



JIT コンパイラによる事前コンパイル

```
import jax
from jax impo

@jax.jit
def h(x1, x2):
    y = 2 * jnp.si
    return y

nabla_h = jax.jit(jax.grad(h, (0, 1)))

plot3d((-jnp.pi, jnp.pi), (-jnp.pi, jnp.pi), h, nabla_h)
```

直後の関数に事前コンパイルを適用

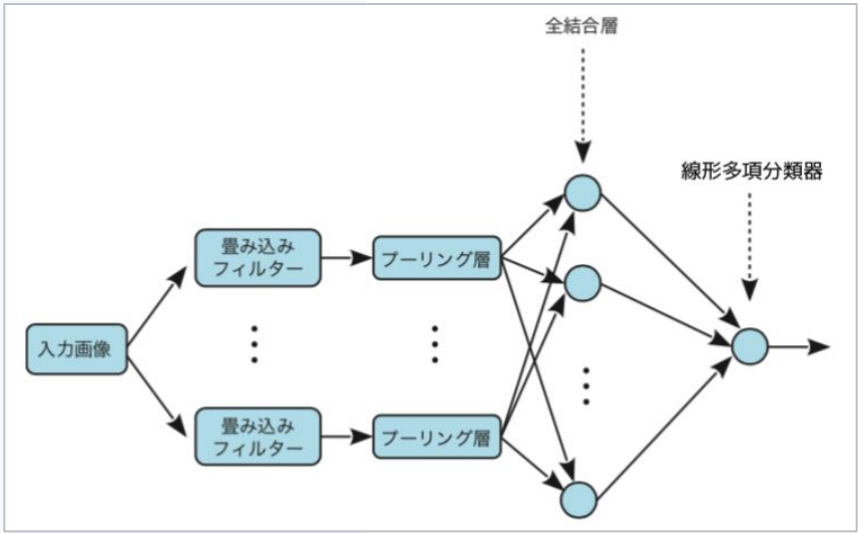
引数で指定した関数事前コンパイルを適用

Flax / Optax 入門

Flax によるニューラルネットワークの定義

```
class SingleLayerCNN(nn.Module):
    @nn.compact
    def __call__(self, x, get_logits=False):
        x = x.reshape([-1, 28, 28, 1])
        x = nn.Conv(features=16, kernel_size=(5, 5))(x)
        x = nn.relu(x)
        x = nn.max_pool(x, window_shape=(2, 2),
                        strides=(2, 2))

        x = x.reshape([x.shape[0], -1]) # Flatten
        x = nn.Dense(features=1024)(x)
        x = nn.relu(x)

        x = nn.Dense(features=10)(x)
        if get_logits:
            return x
        x = nn.softmax(x)
        return x
```

条件分岐を記述可能



誤差関数と学習ステップを個別に関数として実装

誤差関数

```
@jax.jit
def loss_fn(params, state, inputs, labels):
    logits = state.apply_fn({'params': params},
                            inputs, get_logits=True)
    loss = optax.softmax_cross_entropy(logits, labels).mean()
    return loss
```

モデルの予測結果（ロジットの値）を取得

誤差関数（クロスエントロピーを計算）

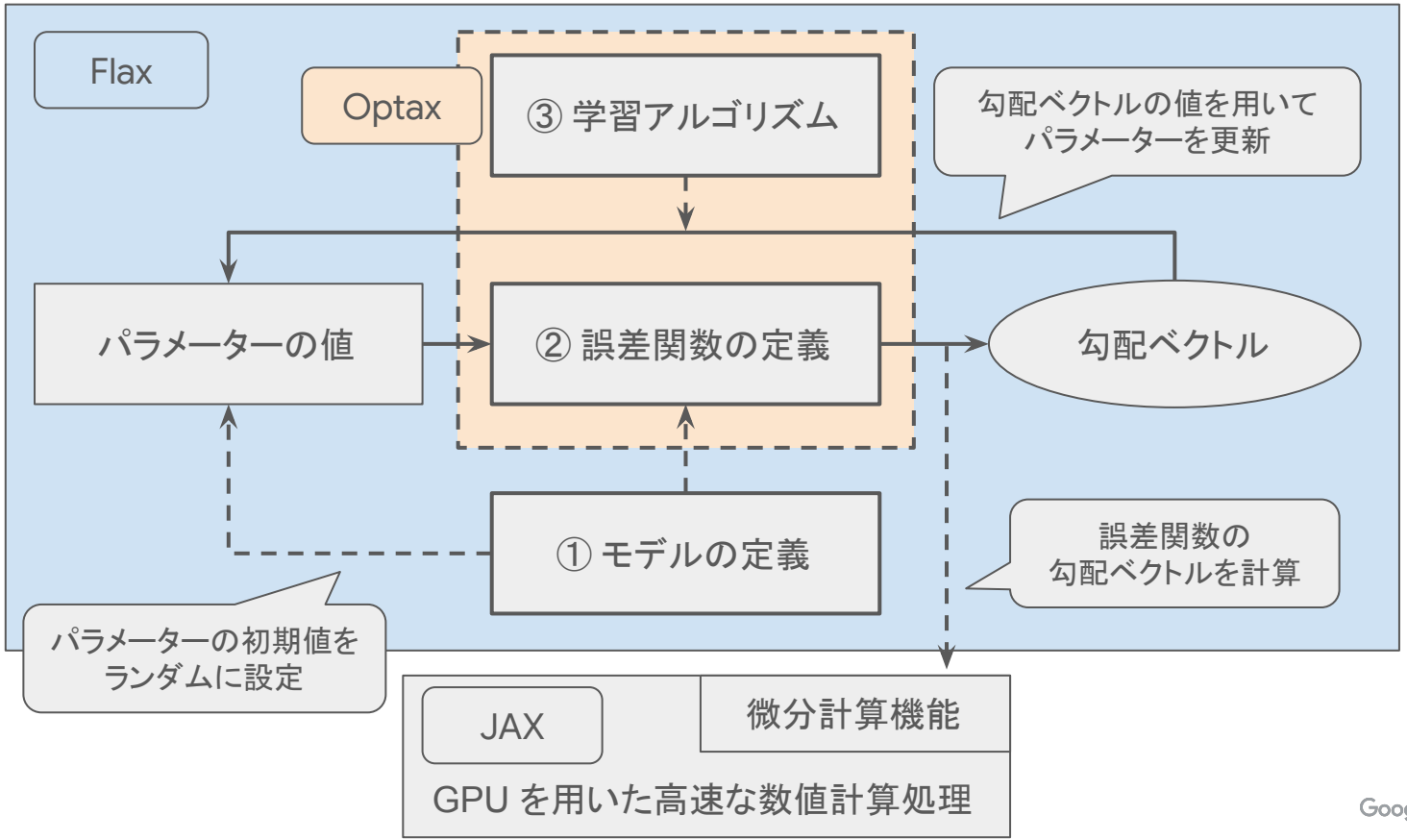
1回の学習ステップ

```
@jax.jit
def train_step(state, inputs, labels):
    loss, grads = jax.value_and_grad(loss_fn)(
        state.params, state, inputs, labels)
    new_state = state.apply_gradients(grads=grads)
    return new_state, loss
```

勾配ベクトルを計算

勾配降下法のアルゴリズムで
パラメーターを（1回だけ）修正

JAX / Flax / Optax の役割分担



JAX / Flax のメリット（個人の感想）

- 誤差関数や学習ステップを自分で関数として実装する必要がある
  - Keras のように「fit() メソッドで一発！」というわけではない
- 誤差関数や学習ステップの中身がブラックボックスにならないので、細かなチューニングや「ちょっと凝った独自の実装」が手軽にできる
  - 定番のモデルを使った実務用途よりは、さまざまなチューニングや実装上の工夫を試す研究・開発用途に適している
- 独自の fit() メソッドをモジュール化して使い回すこともちろん可能
  - 学習状態を管理するTrainState オブジェクトなど、Flax が提供する機能を活用

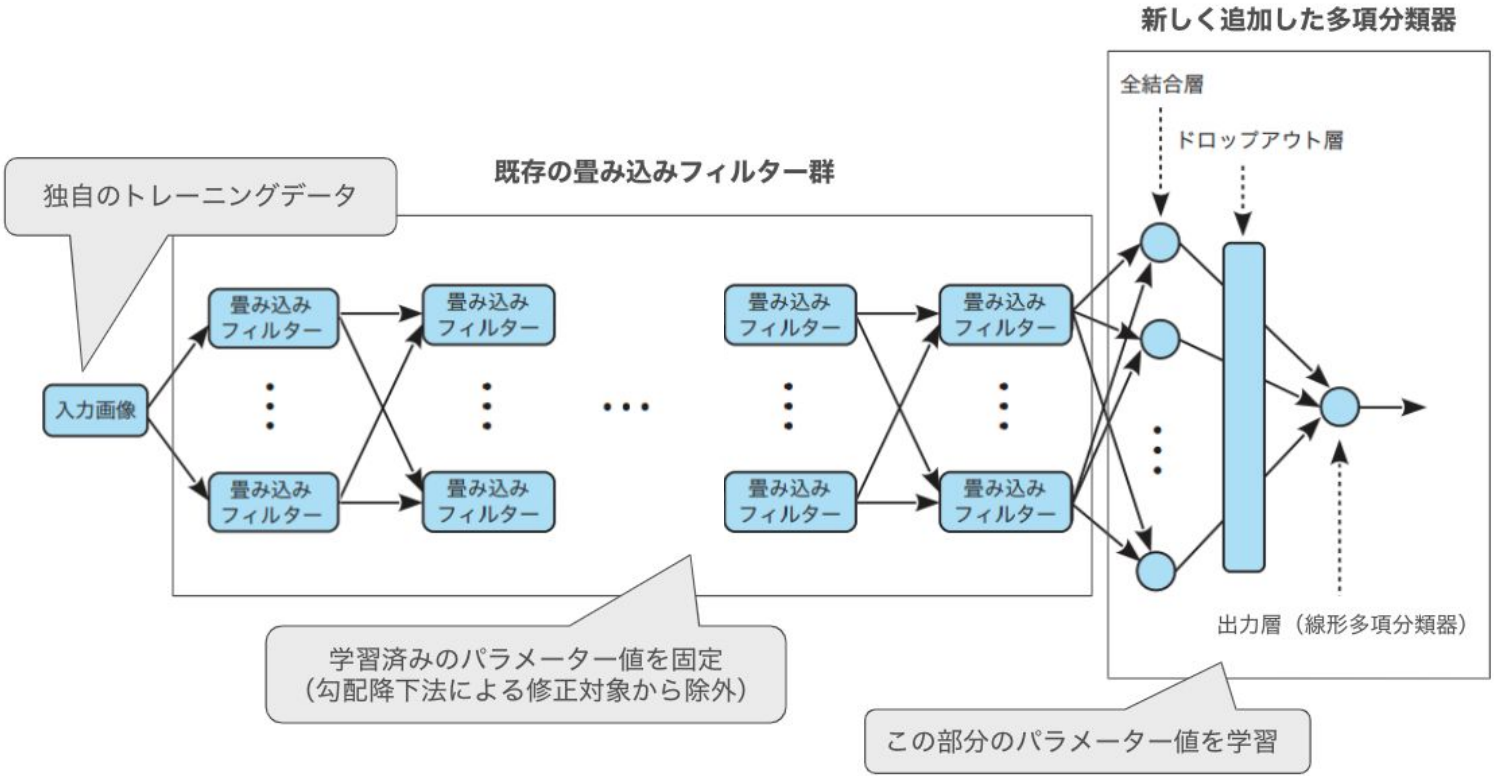
JAX / Flax のその他の特徴

- モデルのオブジェクトは学習中のパラメーター値を含まない
  - パラメーター値は、ディクショナリーで管理
  - パラメーターの一部を書き換えたり、学習済みのパラメーター値を取り出して流用するなどが容易にできる

```
@jax.jit
def loss_fn(params, state, inputs, labels):
    logits = state.apply_fn({'params': params},
                            inputs, get_logits=True)
    loss = optax.softmax_cross_entropy(logits, labels).mean()
    return loss
```

モデルを呼び出す際に
パラメーター値を入力

JAX / Flax による転移学習の例

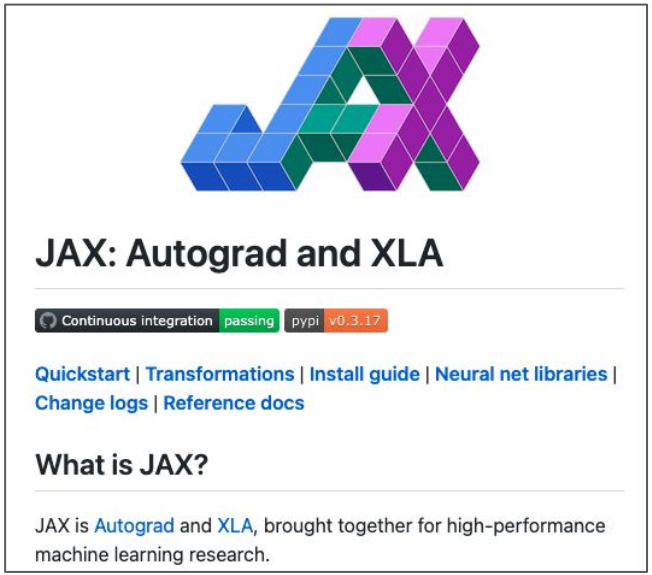


https://gist.github.com/enakai00/a8589d42aa89e2ba005ef9a247296bfc

まとめ（復習）

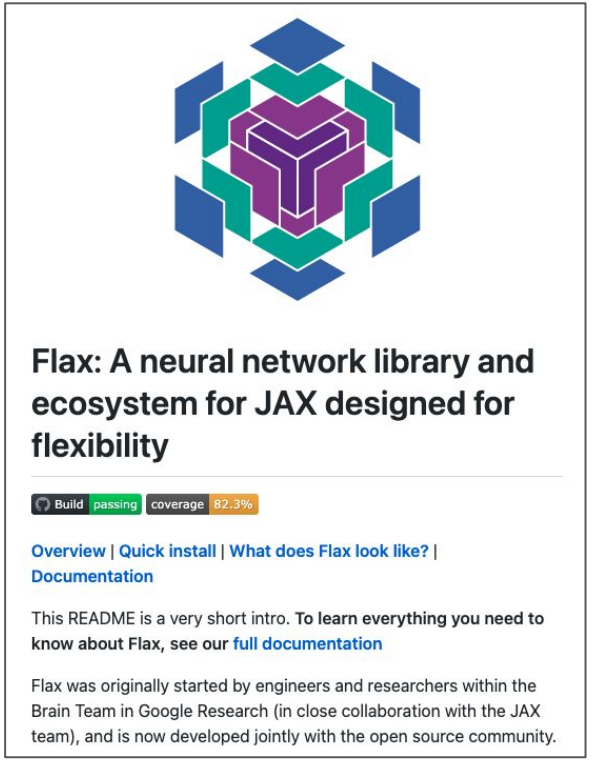
JAX とは？

- Google の AI 研究チーム（Google Brain）が開発した数値計算ライブラリー
- 機械学習のベースとなる計算処理を GPU / TPU で高速に実行可能
- NumPy とほぼ同じ関数（メソッド）を用意しており、NumPy を知っていればすぐに使える
- JAX 独自の機能の例
  - JIT コンパイラ: Python で定義した関数を GPU / TPU での計算に最適化されたバイナリーに事前コンパイル
  - 自動微分 : Python で定義した関数の微分（勾配ベクトル）を計算

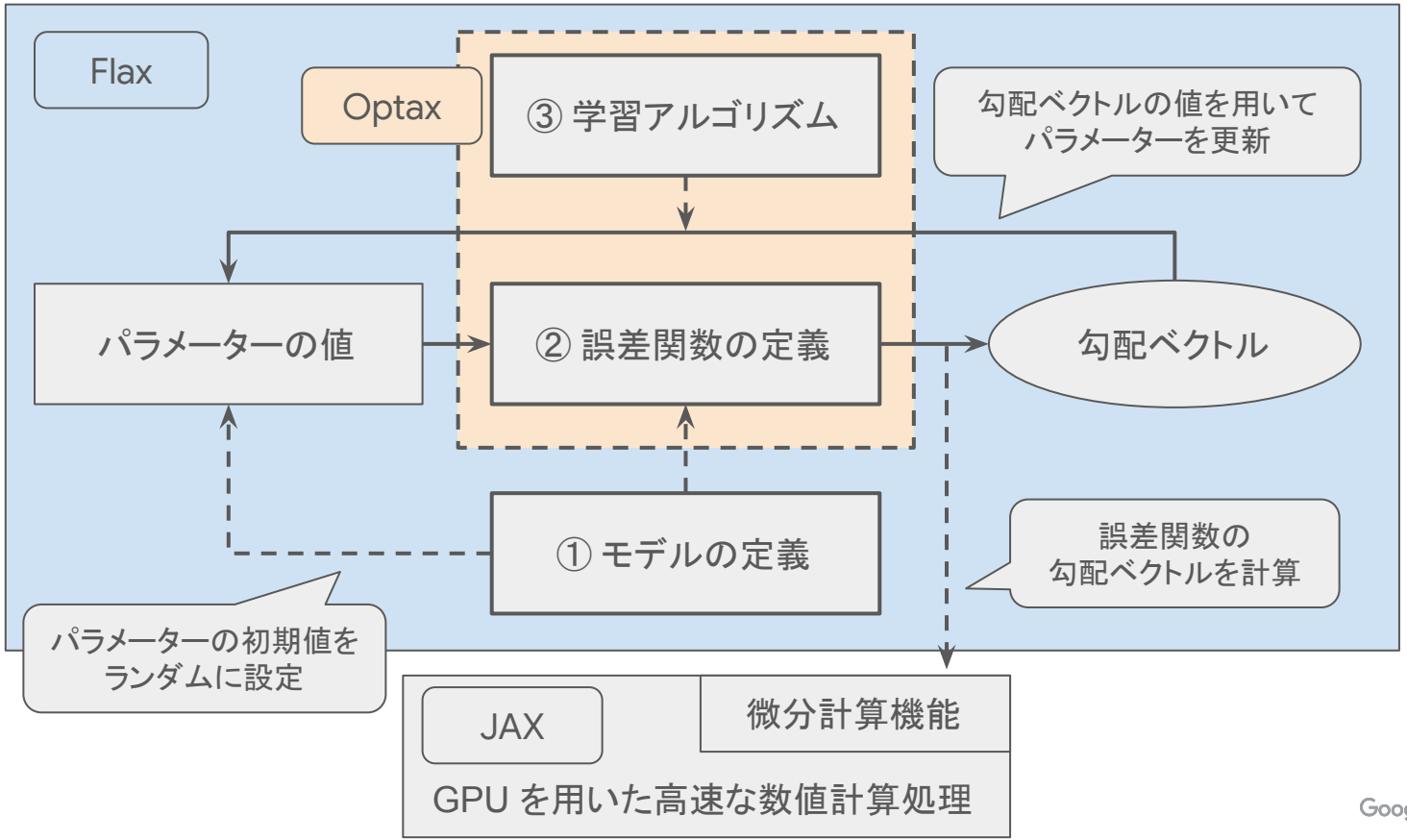


JAX は上位ライブラリーと組み合わせて使用

- JAX はあくまでも数値計算ライブラリーなので、それだけでは機械学習の処理（ニューラルネットワークの定義、勾配降下法による学習処理など）はできない
- さまざまな上位ライブラリーと組み合わせて使用する
  - Flax：ニューラルネットワークの構築と学習プロセスの管理機能を提供
  - Optax：誤差関数や勾配降下法のアルゴリズムをモジュールとして提供
  - などなど
- 今回は、JAX / Flax / Optax の組み合わせ例を紹介



JAX / Flax / Optax の役割分担



Thank you.

Google Cloud