ニューラルネットワークを用いた AIの格闘ゲームへの組み込み

泊　久信
hisanobu.tomari#3675
株式会社SNK ゲーム事業本部 R&D

The Future Is Now
SNK®

hisanobu.tomari#3675

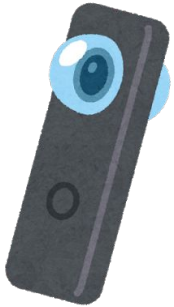
SNK

スライドを公開しています

今すぐダウンロード

https://bit.ly/gcc2019tomari

写真撮影OK



SNS投稿OK

あらすじ

- はじめに
- 既存研究
- 実装
  - プロトタイプ
  - 本番
- 大変だったところ
- 結論

はじめに

「AI」が流行っている

- 画像認識、翻訳、…
- 処理速度の高速化、ニューラルネットの発展が原因

ゲームに組み込んで「遊べる」ようにしたい

- 機械学習の動作を実感できたら楽しい

私達のゲームでやったこと

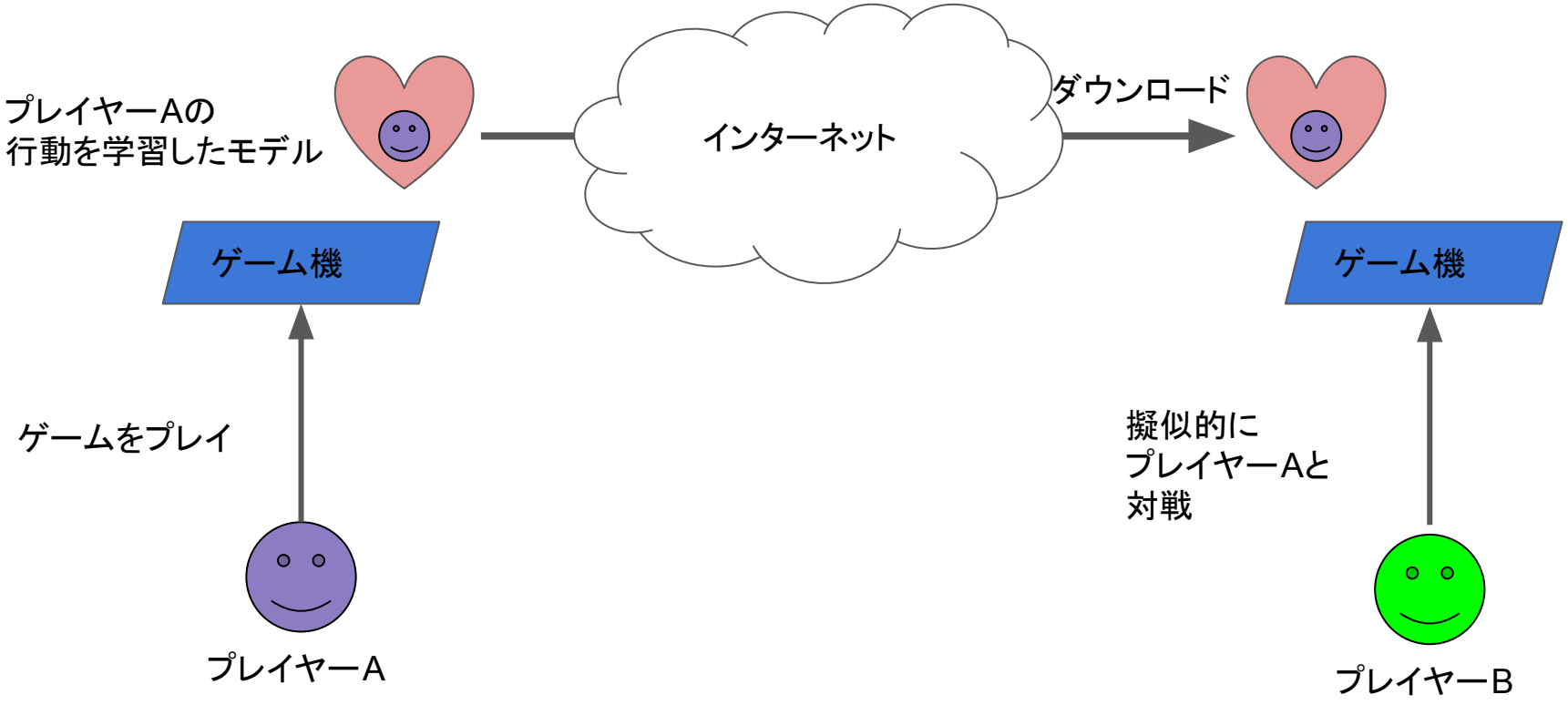
ニューラルネットをゲーム本体に組み込み

機械学習をおもちゃとして遊べるようにしてみる

ゲーム中にリアルタイムで学習

- ゲーム機の上で動かすため、できる限り小さい構成で動くようにした
- プレイヤーの短いプレイ時間からも効率的に学習できるよう工夫

やりたいこと！！


プレイヤーAの
行動を学習したモデル
インターネット
ダウンロード
ゲーム機
ゲーム機
ゲームをプレイ
擬似的に
プレイヤーAと
対戦
プレイヤーA
プレイヤーB

既存のゲームを題材にした機械学習

OpenAI Five (DOTA 2のボット)

- DOTA 2とはSteamのBot APIで接続
- 長時間かけて多数の経験を積んで学習を行う
  - 128,000 CPUコア
  - 256 GPU

	OPENAI 1V1 BOT	OPENAI FIVE
CPUs	60,000 CPU cores on Azure	128,000 preemptible CPU cores on GCP
GPUs	256 K80 GPUs on Azure	256 P100 GPUs on GCP
Experience collected	~300 years per day	~180 years per day (~900 years per day counting each hero separately)
Size of observation	~3.3 kB	~36.8 kB
Observations per second of gameplay	10	7.5
Batch size	8,388,608 observations	1,048,576 observations
Batches per minute	~20	~60

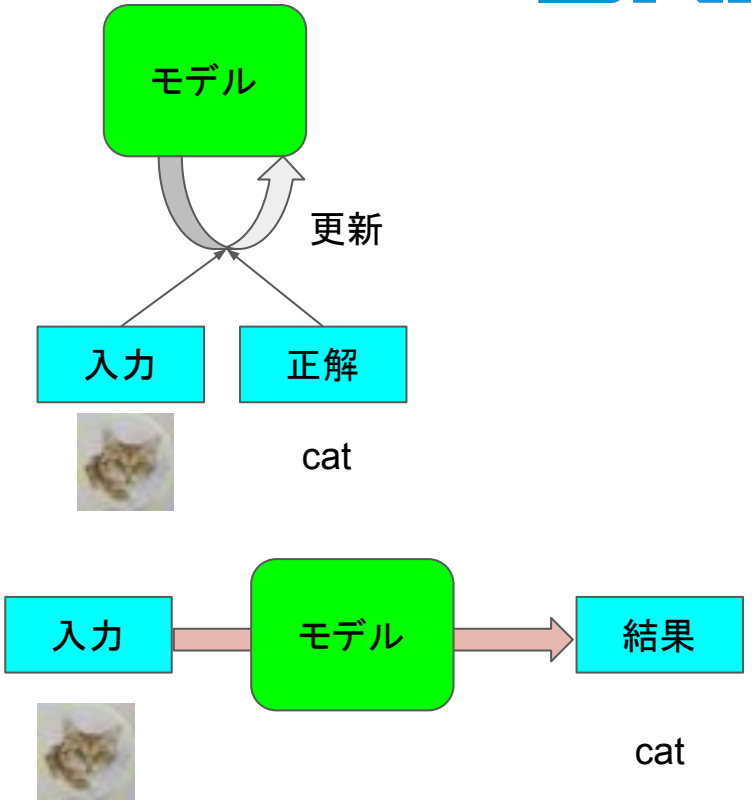
ニューラルネットの大枠

学習

- 訓練データをモデルに反映
- 計算量が多い

推論

- モデルを使って出力を計算
- 計算量は学習処理に比べて少ない



AIの歴史

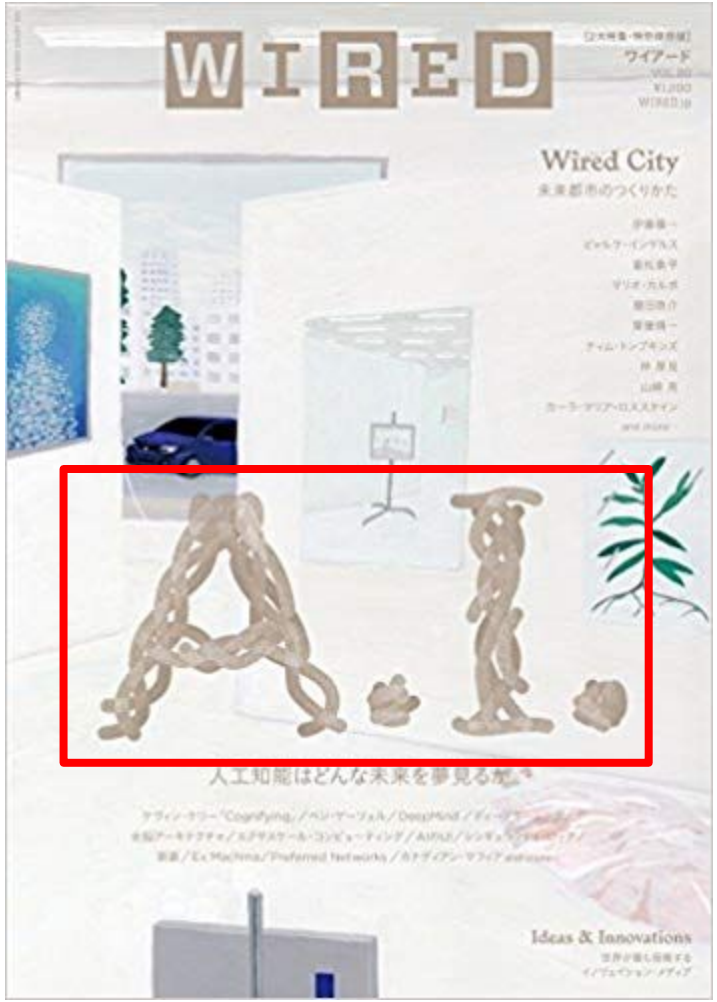
BYTE
THE SMALL SYSTEMS JOURNAL
APRIL 1985 VOL. 10, NO. 4
$3.50 IN UNITED STATES
$4.25 IN CANADA / £2.10 IN U.K.
A McGRAW-HILL PUBLICATION
0360-5280
ARTIFICIAL
INTELLIGENCE

AIの歴史

Is Artificial Intelligence Dead?
PAGE 236
BYTE
JANUARY 1991
A McGRAW-HILL PUBLICATION
Toshiba's New T1000LE
PAGE 127

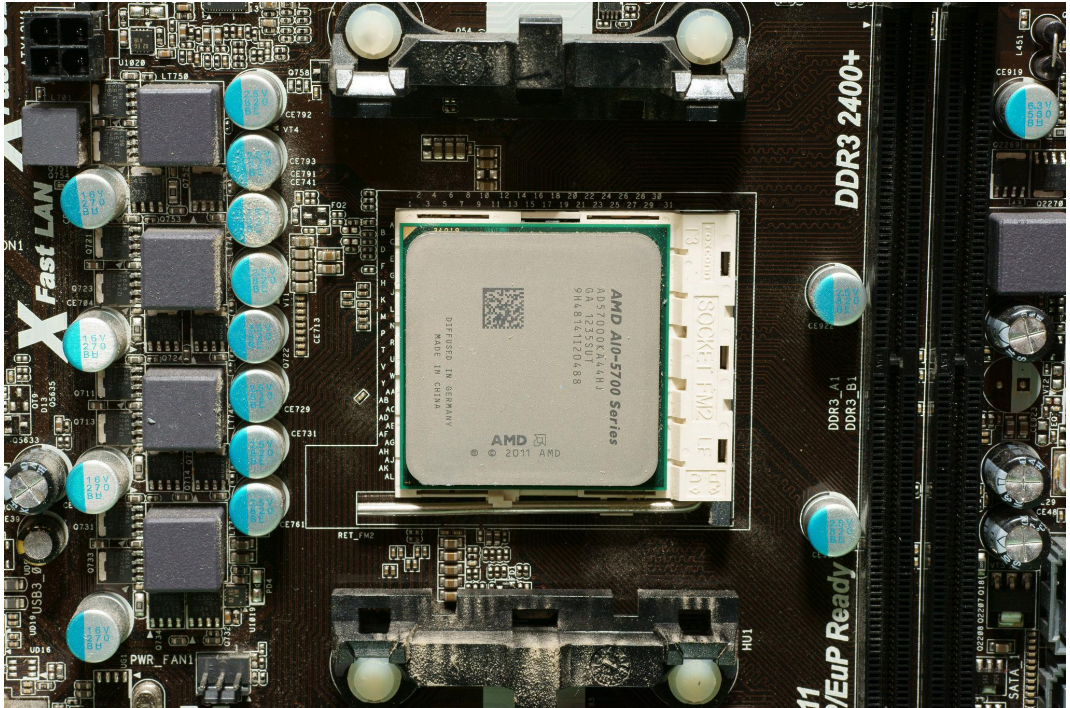
AIの歴史

A.D.2015



AIの歴史

CPUの分岐予測にパーセプトロンを使った例



AIの歴史

歴史は繰り返す？

並列推論マシンPIM

http://www.jaist.ac.jp/iscenter-/mpc/old-machines/pim/

Google Cloud TPU

https://cloud.google.com/blog/products/gcp/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

ニューラルネットいろいろ CNNとRNN

CNN

- 空間的に広がりのあるデータを扱うのが得意
- 最近は時系列のデータも扱えるとの研究もあり

RNN

- 時系列のデータを扱うのが得意
- 今回つかったのはこちらの系統

RNNいろいろ

LSTM

- 因果関係に時間的な隔たりがあってもうまくいく（こともある）

GRU

- LSTMと似た構造と性質を持つ。構造が単純化されている。

Clockwork RNN

- 時間的にさらに長いデータでも扱えることがある。
- 試してみた結果、今回の使い方ではいちばん良かった

ゲーム内容の概要

サムライスピリッツ

- 対戦格闘ゲーム
- バトルは自分と相手の１対１
- 相手のライフを0にすれば勝ち

格闘ゲームでのコマンド入力と技

- 順番に入力することで技が発動するキー入力をコマンドという

- 技によって異なる性格があり、トレードオフを生む
  - 相手から受けるダメージを軽減する（ガード）
  - 相手のライフを大きく減らすが、ガードできない時間が長くスキができる
  - 遠くに・近くにダメージを与える
- 通常プレイ時、プレイヤーは画面の状態をみて、適切なキーを時系列で入力していく

コマンド入力のひみつ

- コマンドはゲームで持っているテーブルに保存されている
- 実際のキー入力からコマンドへのマッチングは探索で行う
  - 途中に不要なキー入力があってもマッチするようになっている

- テーブルにはコマンドとともに優先度が保持される
  - 複数のコマンドが同時にマッチした場合、優先度の高いコマンドを受理
- コマンドごとに、キャラクターの状態に依存し受理されないタイミングがある

機械学習の組み込み方針

- プレイヤーの動きを模倣するモデルと遊べるようにする！
  - プレイヤーは「ゲームステート」を入力として、「押したいキー」を出力する箱とみなす
  - この箱を模倣する

- 機械学習モデルをプレイヤーの行動を用いてゲーム機の上でリアルタイムに作る
  - ゲーム機の上でリアルタイムにモデルを作れるように単純な構成を工夫する
- 作ったモデルをオンラインで共有可能にする

ゲームとしての組み込み方

- ゲームモード「道場」モード
- リーダーボードを利用して、他者のゴーストデータ（学習済みモデル）をダウンロード
- このゴーストと対戦できる

- 通常対戦時（オフライン・オンライン）時に行動履歴を用いて学習モデルをアップデート
- 使用頻度が高いキャラの学習モデルをリーダーボードにアップロード

実装していく話

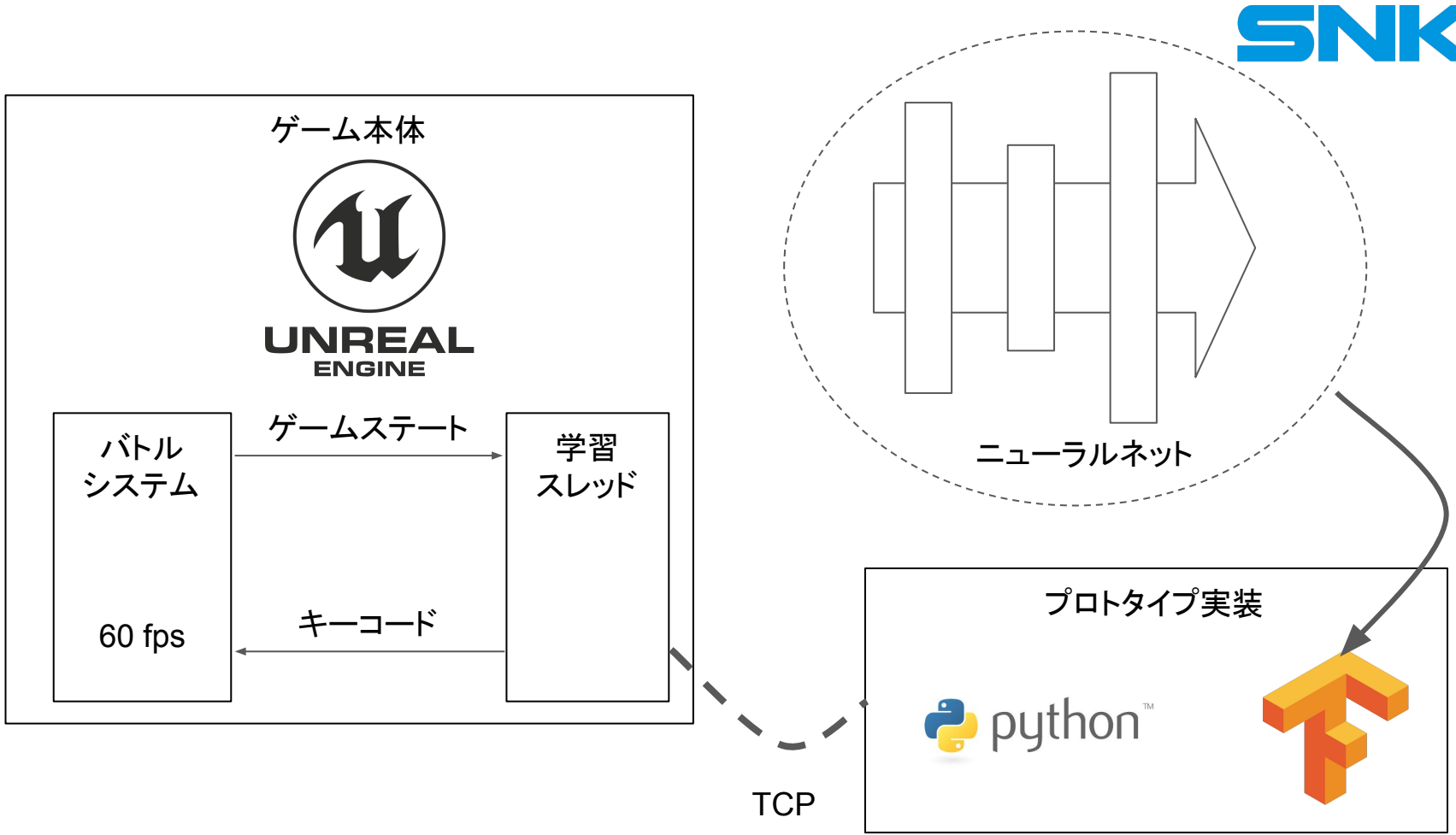
プロトタイプ作成

- 既存の機械学習の実装を、手っ取り早くゲームと接続したい

Tensorflow

- ゲーム本体はUnreal Engine 4 (UE4)で作成されている
- TensorFlow はPython で使うのが流行り → UE4 に直接埋め込むのは手間
  - 特にコンソールでPython を埋め込んでしまうのが困難

UE4 とプロトタイプはTCP接続一本で接続する


SNK
ゲーム本体
UNREAL
ENGINE
バトル
システム
60 fps
ゲームステート
キーコード
学習
スレッド
ニューラルネット
プロトタイプ実装
python
TCP

TCPで機械学習処理を外部に実装する

利点

- ゲーム本体側に影響なく処理を大幅に変えられる
  - ニューラルネットの構成変更
  - 前処理・後処理の変更
- 機械学習処理部でTensorFlow をPythonから使える
  - 機械学習処理部はPython のみで記述

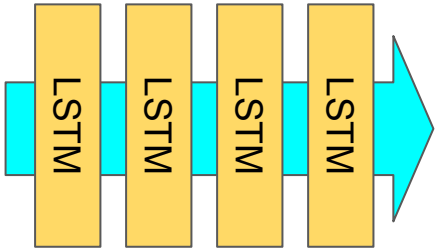
欠点

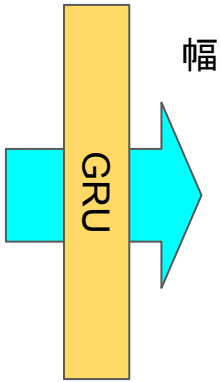
- リリースする前に単一プロセスにするために書き換える必要あり
  - コンソール対応

プロトタイプでいろいろな構成を試す

ニューラルネットの構成を差し替えて試す

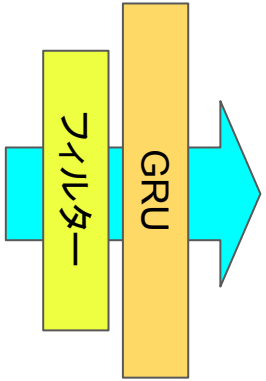
四段構成（学習うまくできない）





幅が広い

アクティベーションを試す



最適化のアルゴリズム

- RMSprop
- Adam
  - 今回はAdam で十分な性能を得ることができた

プロトタイプで試す: 入力と出力

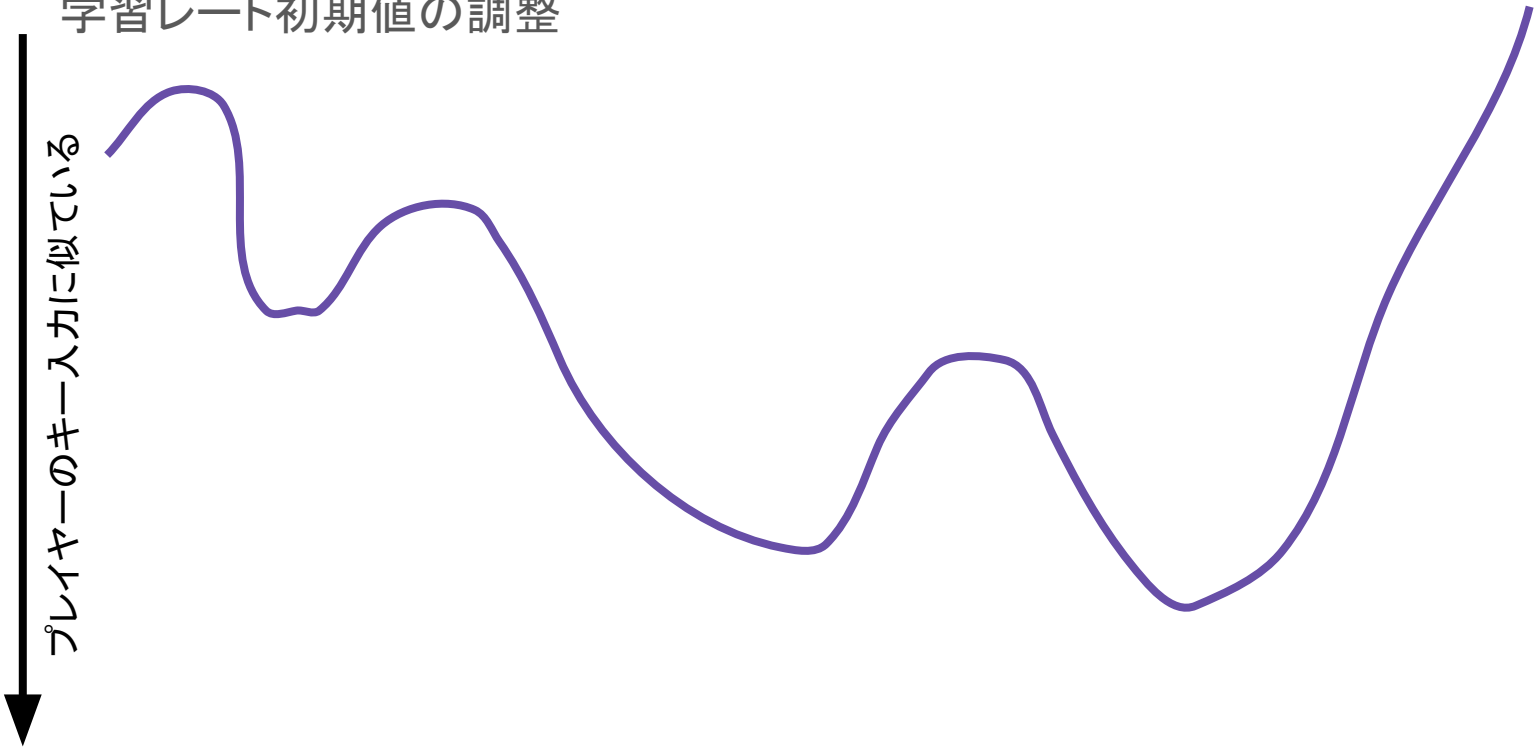
ゲームシステムから手に入れることができるデータを入力

- キャラ位置
- 矩形データー部
- モーションの状態
- ゲージの状態
- 残り時間
- 最終ラウンドか？　など

出力はそれぞれのキー組み合わせをどれくらい押したいかの確率分布

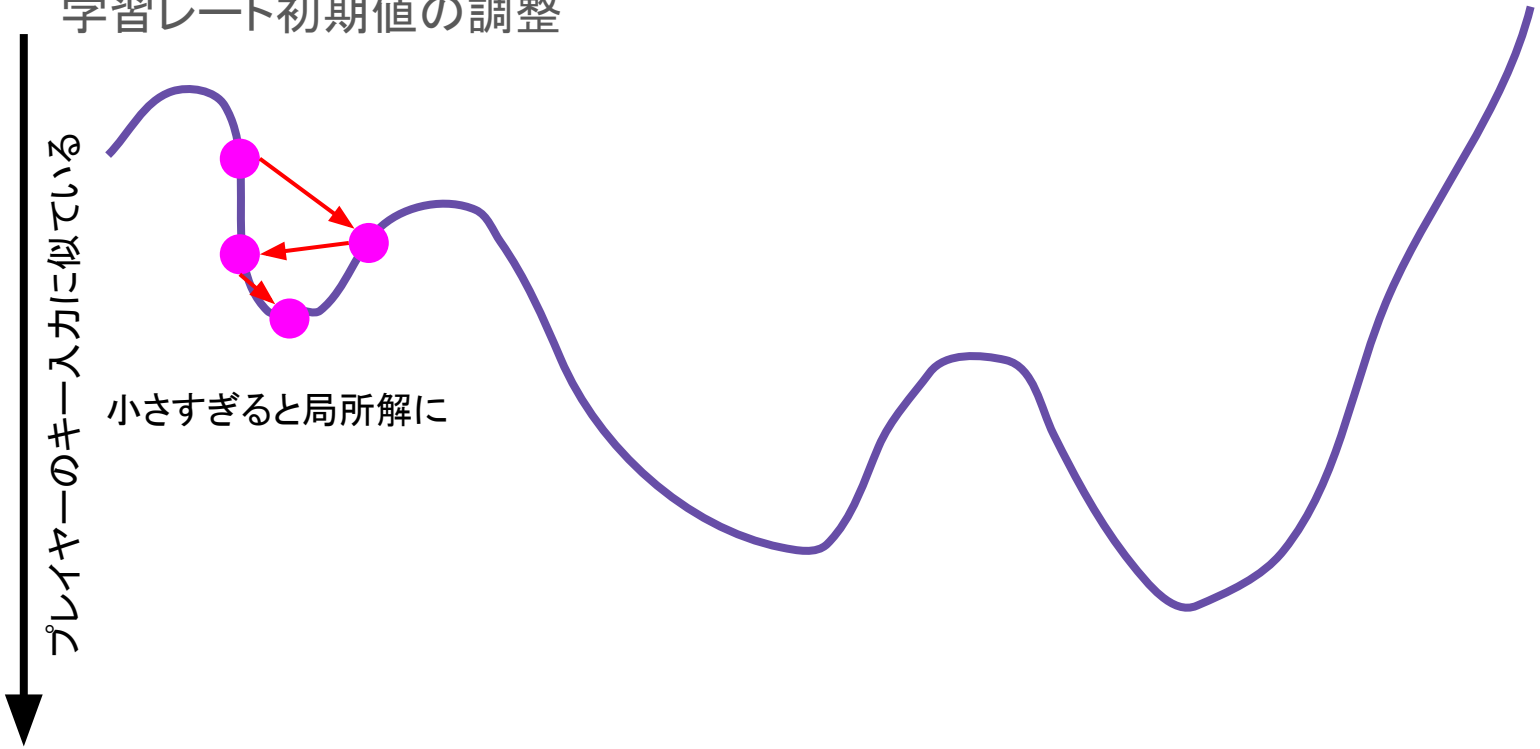
プロトタイプで何をしたか - SGD

学習レート初期値の調整



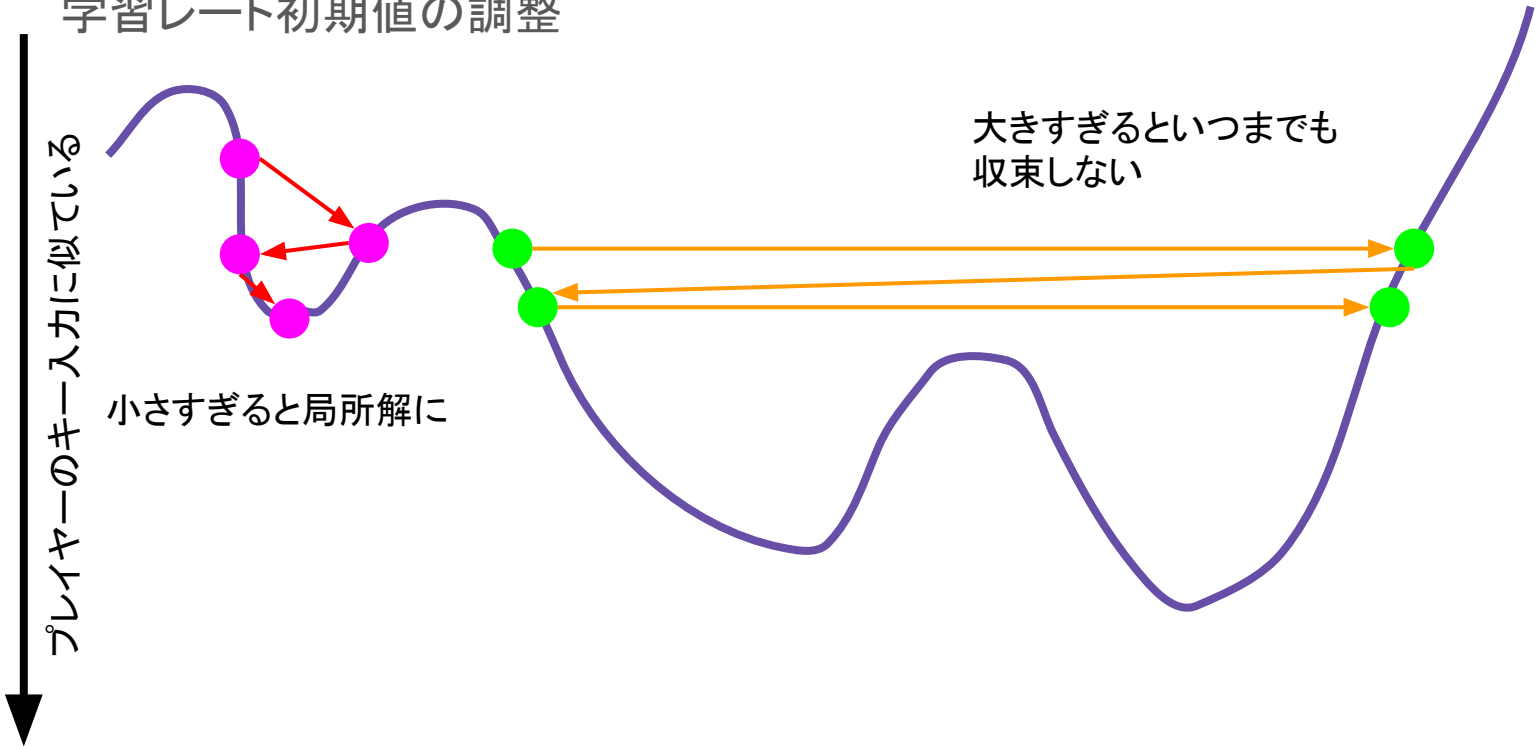
プロトタイプで何をしたか - SGD

学習レート初期値の調整



プロトタイプで何をしたか - SGD

学習レート初期値の調整

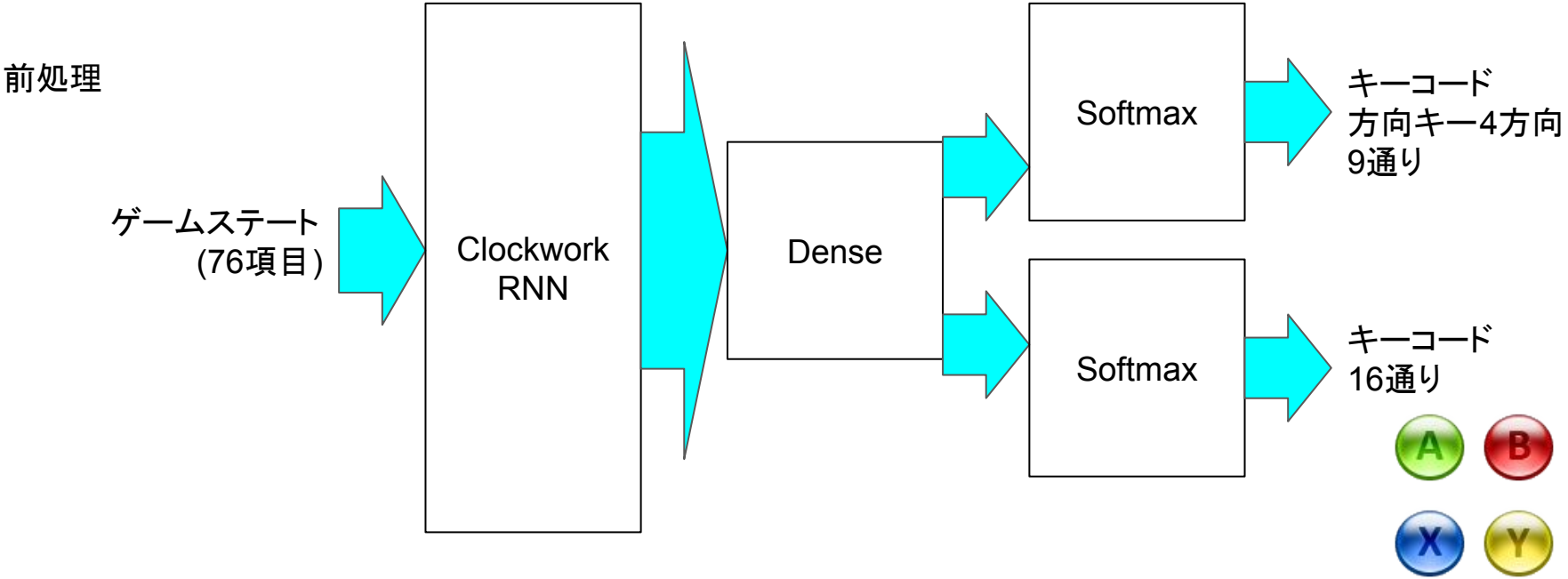


本番用実装

- Tensorflow がやっていてくれた処理を自前で実装する
  - 行列
    - Adam
  - 推論のときの処理
  - 学習のときの処理

- Eigen という行列ライブラリ(C++テンプレート集)を使って実装
- コンパイラ最適化でかなり高速化
  - デバッグビルドで大幅に遅くなりトラブルも 😨
- 非正規化数を捨てるようにするとかなり速くなった (MXCSRレジスタ)
  - _mm_setcsr(_mm_getcsr() | (1<<15)); // FTZ
  - _mm_setcsr(_mm_getcsr() | (1<<6)); // DAZ

ニューラルネット構成


前処理
ゲームステート
(76項目)
Clockwork
RNN
Dense
Softmax
Softmax
キーコード
方向キー4方向
9通り
キーコード
16通り
A
B
X
Y

推論処理の動作

1フレームごとに
キーコードを送る

1フレームごとに
ゲームステートを送る

学習スレッド

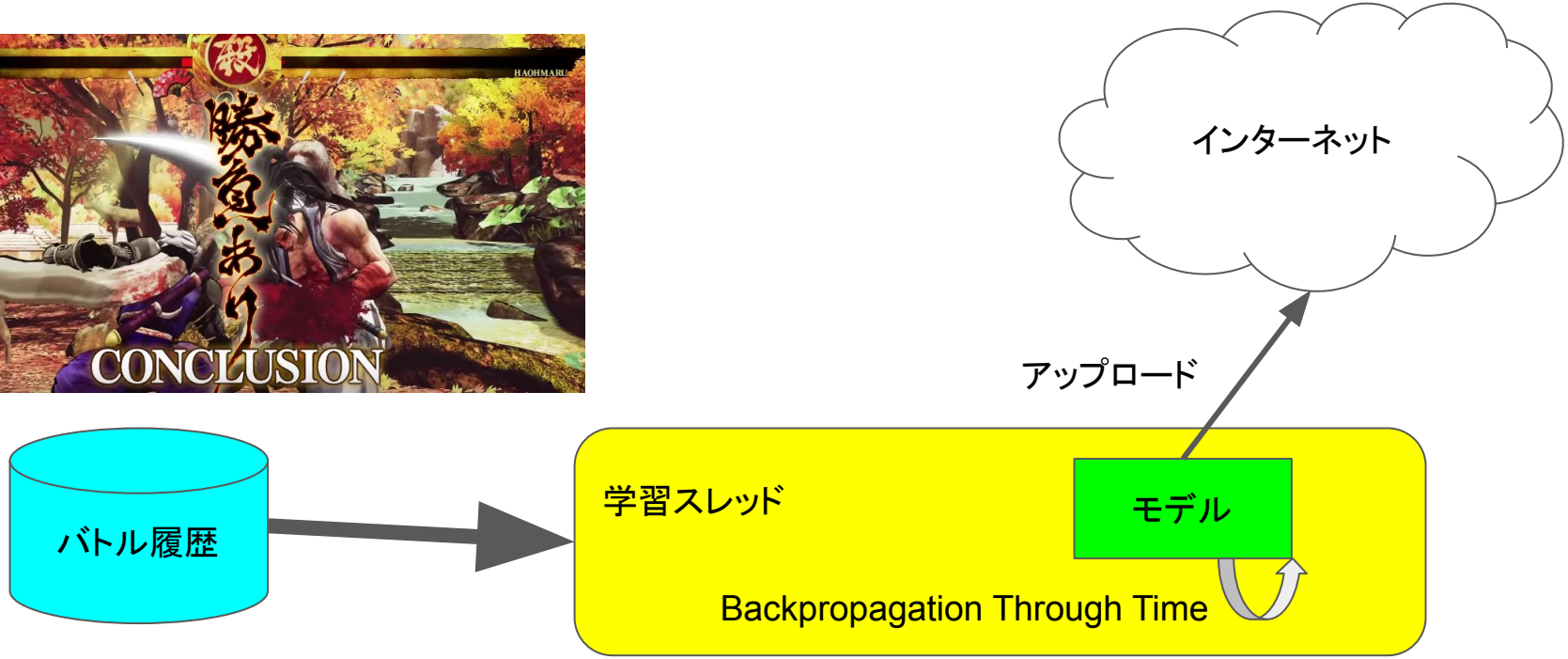
キー入力の生成の工夫

- 学習結果は「押したいキー」の確率分布
- 毎フレーム結果が出てくる

- 学習が不十分な時、どのキーを押すか境界のところでガタガタする
  - ガチャプレイっぽくて不評
- 出力される確率分布が均一に近いときはキーを押さないように

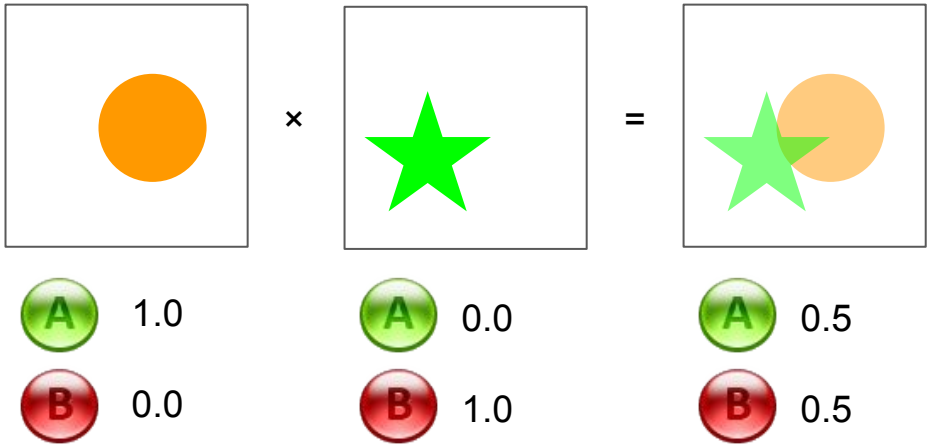
学習処理の動作

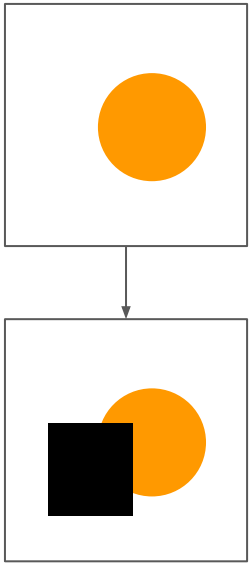
1ラウンド終了ごとに、1ラウンド分のデータを処理



学習の際の工夫

- 画像認識のための学習のテクニックが応用できる部分もあった
  - バッチを重ね合わせて、結果も重ね合わせたものを学習させる
  - 入力の一部分を意図的に隠す（ゲームステートの一部を隠す）
- 単純にゲームステートにローパスフィルタをかける

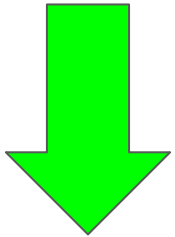




作ってからわかった機械学習の大変さ

人間によるプレイデータを取得する必要がある

- 実際の対戦に近い環境でのプレイ状況をログに取得
  - 人間vs人間



- プランナー２人で1セット50ラウンドくらい対戦してもらっていた
  - ニューラルネットに入力する項目を変更するとやり直してもらう
- 時間がかかるため大変

マスタデータの更新に学習結果が追従しない

- ゲーム開発中、技のコマンドや性質が大きく変わる
- キーコードを出力とするニューラルネットはそれに適応しない

- 人間によるプレイデータを取得し直し

細かい調整が利かない

「ステージの端でキャラが離れているときにジャンプし続けないようにしてほしい」

「この技を確実に使うようにしてほしい。とても大事。」

「もっとガードするようにしてほしい」

ニューラルネットでこれらの要望を直接叶えるのが困難

学習するときに入れるデータを工夫してやる？どうやって工夫する？

できなかったこと

より人間のキー入力に似せた出力をしたかった

- ネットワークで共有できるファイルサイズの制限
- リアルタイムに学習処理を行えるニューラルネットの大きさの制限

学習処理も高速化できるようなエッジ用のアクセラレータの標準搭載を希望

クラウドで学習処理を行う

- 継続的にサービスで収益を上げられるような売り方への転換

結論

- ニューラルネットを用いて、リアルタイムにプレイヤーの行動から学習するシステムをゲームに組み込んだ。
  - ニューラルネットで遊べるようにした
- 工夫と割り切りで現行ハードウェアで実装すること自体は可能。

- より大きなモデルを使うことができれば、精度を上げることができる見込み
  - 処理速度
  - ネットワークで共有できるファイルの大きさ
- 設計段階から機械学習の難しいところを考慮すべき

謝辞

忍耐強いチームの皆さんに感謝

The Future Is Now

SNK®