

---

# 通信・計算機システムと 知的情報処理 (2/4)

情報通信系

工学リテラシー

資料作成  
篠崎、中原、上垣外  
およびTAメンバー

# 取り組み内容

---

- 1回目
  - ニューラルネットの原理
  - TSUBAMEアカウント作成
- 2回目
  - ニューラルネット課題レシピの説明
  - スーパーコンピュータTSUBAME3.0の利用法
- 3回目
  - 課題レシピの動作確認
  - 課題レシピの改造
- 4回目
  - TSUBAME上での実験の続き
  - レポートの作成

# ニューラルネットレシピ

---

- ニューラルネットの学習と評価を簡単に体験してみるための3分クッキング的なスクリプト(レシピ)を用意しました
- 各レシピは、学習データの準備、ニューラルネットの学習および評価までを行います
- 本講義では、東工大のTSUBAMEスーパーコンピュータ上で実行します

# 課題実験の方針

目的: ニューラルネットの学習と評価の大枠を理解する

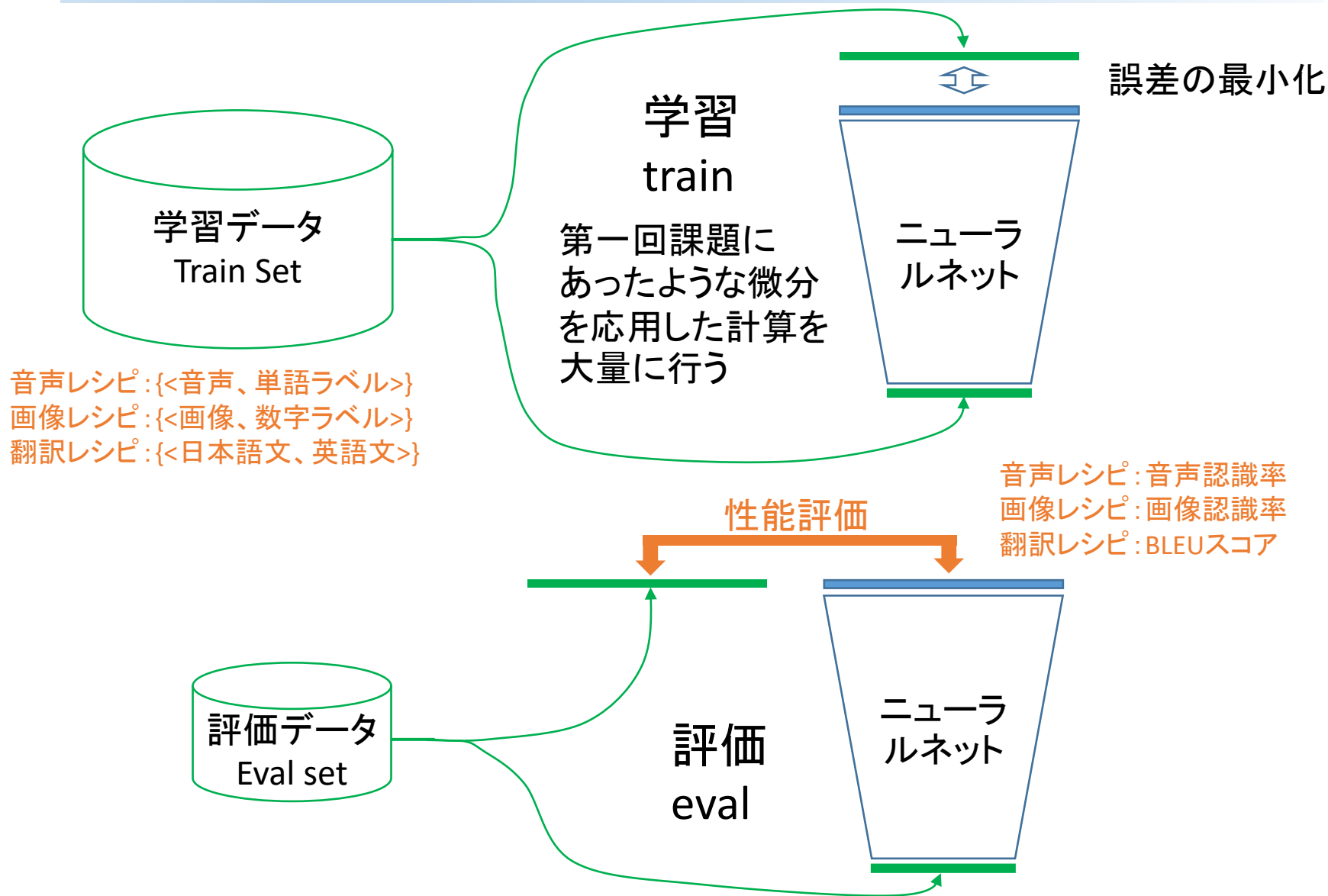
- 理解してほしい内容

- 学習データを用いてニューラルネットを学習する
- 評価データを用いて性能評価を行う
- ニューラルネットの学習には沢山の学習データと大規模な計算が必要
- 各レシピにおける評価尺度のざっくりとした定義
  - どういう意味合いがあるのか
  - 大きい方がよいのか小さい方がよいのか

- 読み飛ばしてOKな内容

- 各レシピのネットワーク構造
- 評価尺度の正確な定義
- プログラムコードの詳細

# レシピ共通の枠組み



# 音声認識レシピ

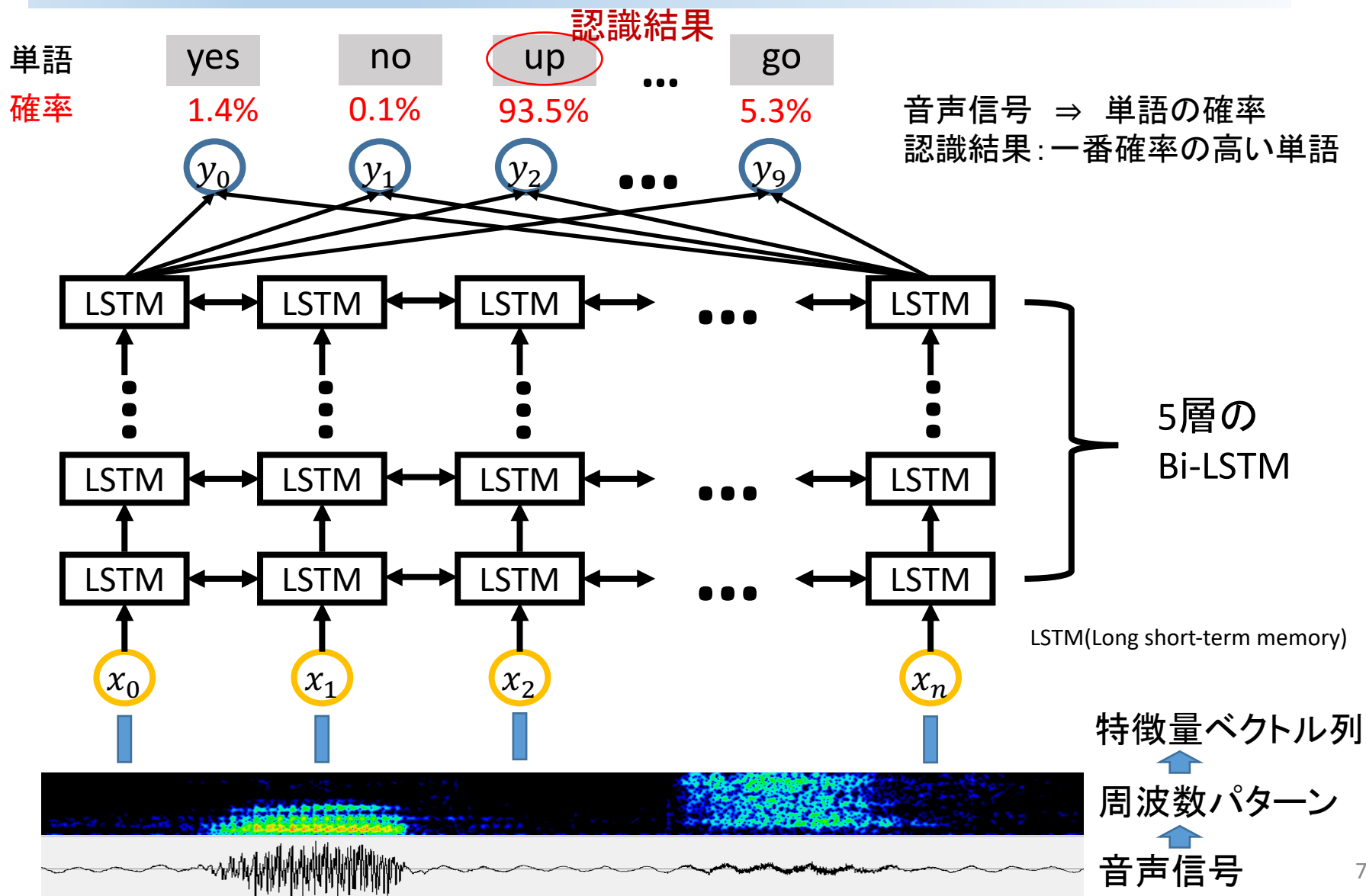
- Googleが公開しているSpeech Commands Datasetを用いて、音声コマンドを認識するニューラルネットを作成します
- 認識の対象とするのは、10単語 (yes, no, up, down, left, right, on, off, stop, go)です

参考:

<https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html>

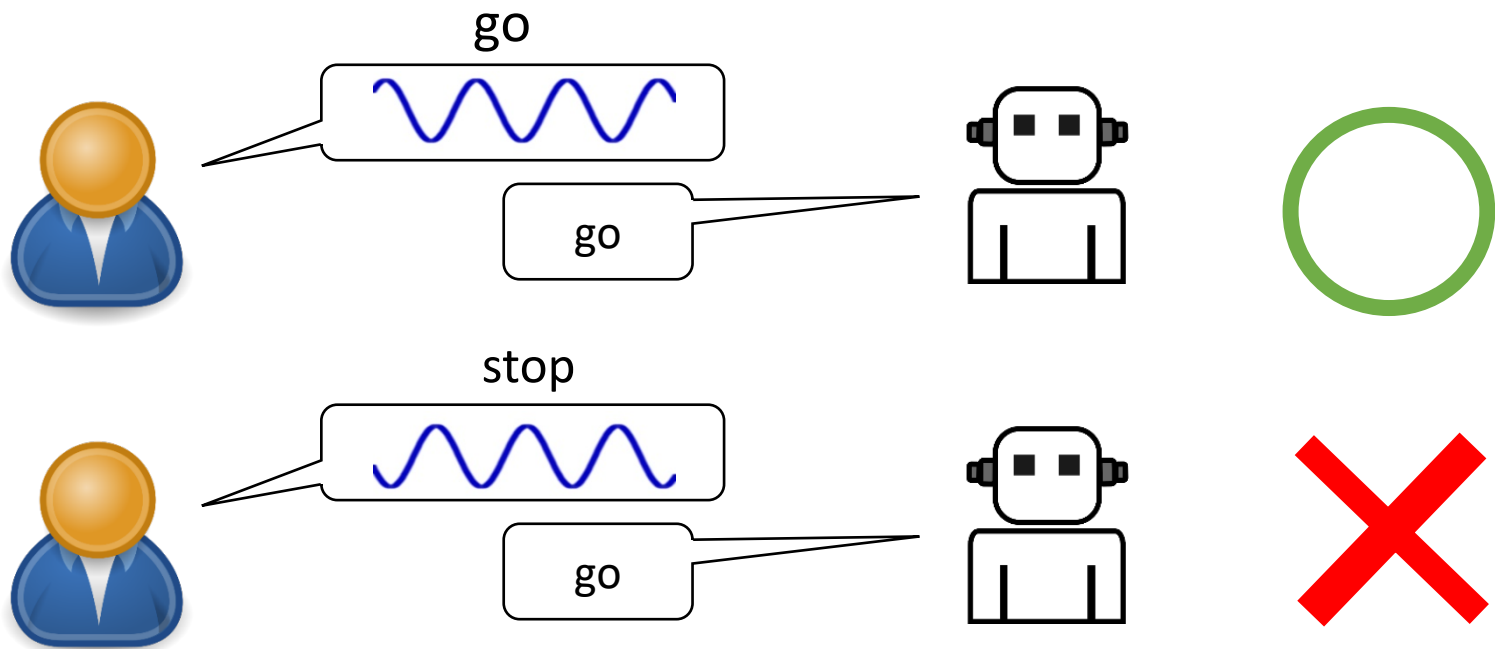


# 音声認識ニューラルネット



# 性能評価尺度(音声認識)

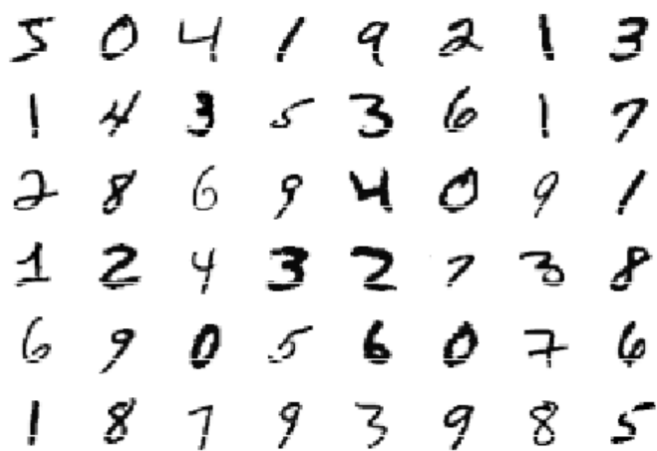
$$\text{認識率} = \frac{\text{認識に成功したコマンド数}}{\text{テストに用いたコマンド数}}$$





# 画像認識レシピ

- 3層ニューラルネットワークを用いて  
手書き文字画像・ファッション画像の分類を行う
- 学習・評価データセット
  - MNIST:
    - 0から9までの手書き画像 (28 × 28ピクセル, モノクロ)
  - Fashion MNIST:
    - 10種類のファッション画像 (28 × 28ピクセル, モノクロ)



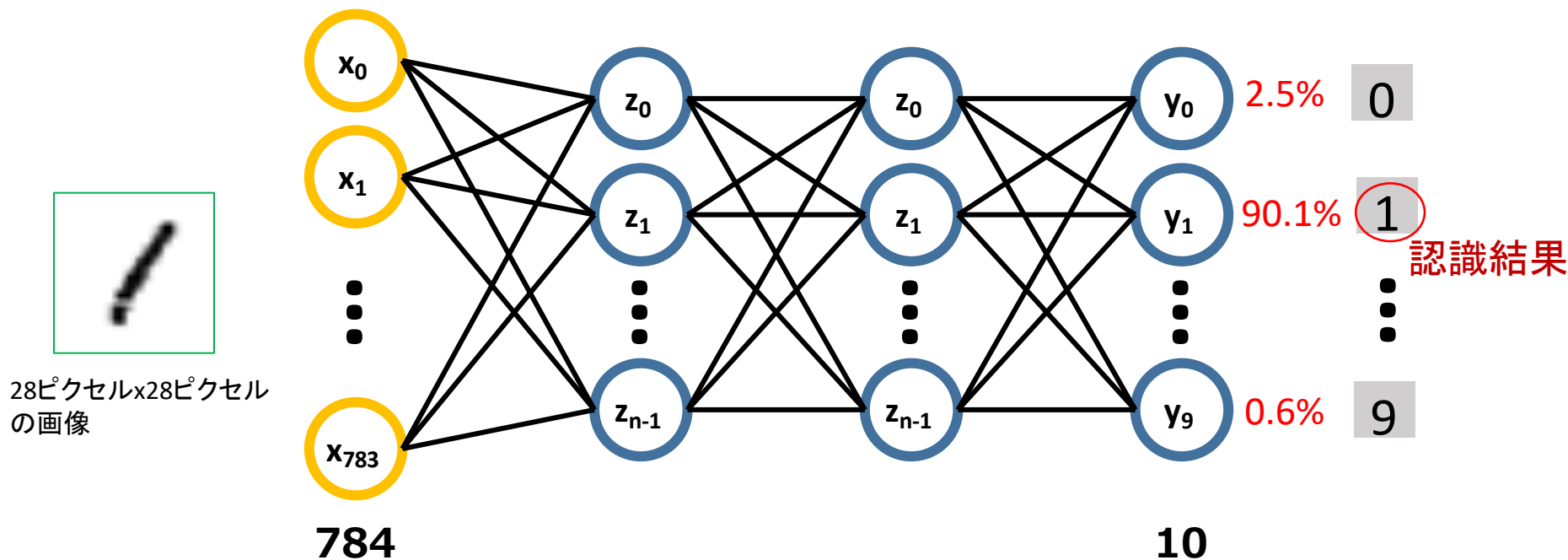
MNISTの例



Fashion MNISTの例

# 画像認識ニューラルネット

- 3層ニューラルネットワーク
- 入出力ニューロンが全て結合しているフル結合構造



# 性能評価尺度(画像認識)

- 認識率=  $\frac{\text{分類に成功した画像数}}{\text{テストに用いた画像数}}$
- 参考: 認識結果の詳細確認
  - レシピの `show_examples(モデル, テスト画像, gpu番号)` を実行

No.45  
Answer:5  
Predict:5



No.46  
Answer:1  
Predict:1



No.47  
Answer:2  
Predict:2



No.48  
Answer:4  
Predict:4



No.49  
Answer:4  
Predict:4



No.50  
Answer:6  
Predict:6



No.51  
Answer:3  
Predict:3



No.52  
Answer:5  
Predict:5



No.53  
Answer:5  
Predict:5



No.54  
Answer:6  
Predict:6



# 機械翻訳レシピ

- **Sequence-to-Sequence (Seq2Seq)**を用いて日英・英日機械翻訳の学習と評価を行う
- 学習と評価には**京都フリー翻訳タスク**を使用
  - <http://www.phontron.com/kftt/index-ja.html>
  - 京都に関連する観光と歴史に関する記述を対象としたデータセット
  - 日本語と英語の文が対になっている

## 京都フリー翻訳タスクに含まれる翻訳対の例

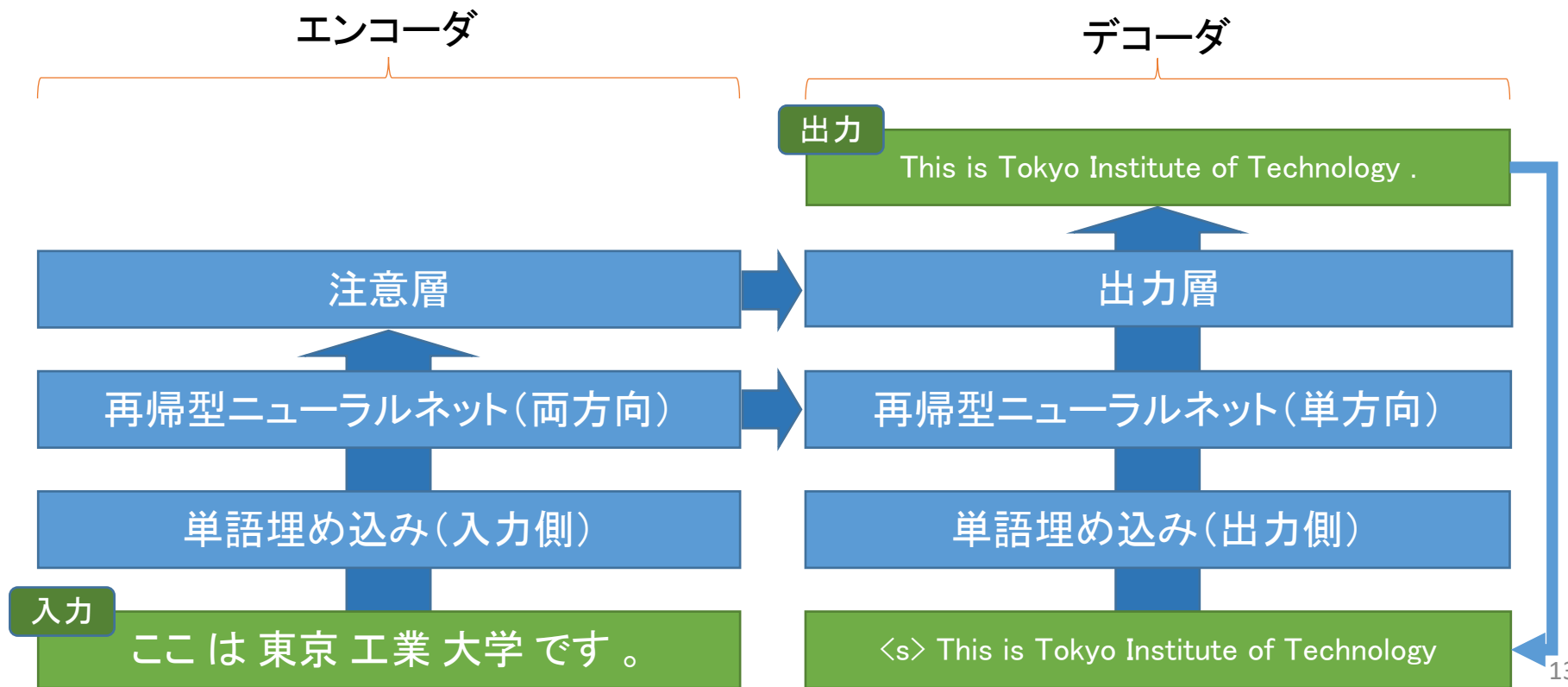
平成以降、安倍晴明のブームが起こり、全国から参拝者が訪れるようになった。

During the Heisei era, Seimei ABE became extremely popular and worshipers have come to visit the shrine from all over the country.

# Seq2Seqによる機械翻訳

- **Sequence-to-Sequence (Seq2Seq)**

- 単語列をベクトルに変換するエンコーダと、  
ベクトルを単語列に変換するデコーダにより構成
- 翻訳では原言語をベクトル表現を介し目的言語に変換



# 性能評価尺度（機械翻訳）

- BLEUスコアで評価
  - 参照訳と翻訳結果中の単語列の一致率により評価
  - 一致率が高ければスコアが大きくなる
  - 人手評価との相関が高いことが知られている

$$\text{BLEUスコア} = \text{長さペナルティ} \times \text{参照訳との一致率}$$

## 参考

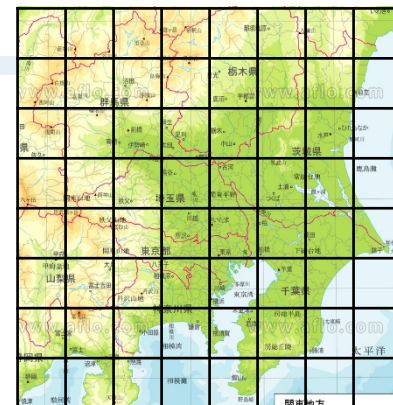
**長さペナルティ:** 参照訳に対する翻訳結果の長さの比率。  
1を超える場合は1とする

**参照訳との一致率:** 翻訳結果に含まれるN単語の連なりの内、  
実際に参照訳に含まれるものの割合  
翻訳結果に含まれるN単語の連なりが参照訳よりも多い場合は超過する分については無視して扱う

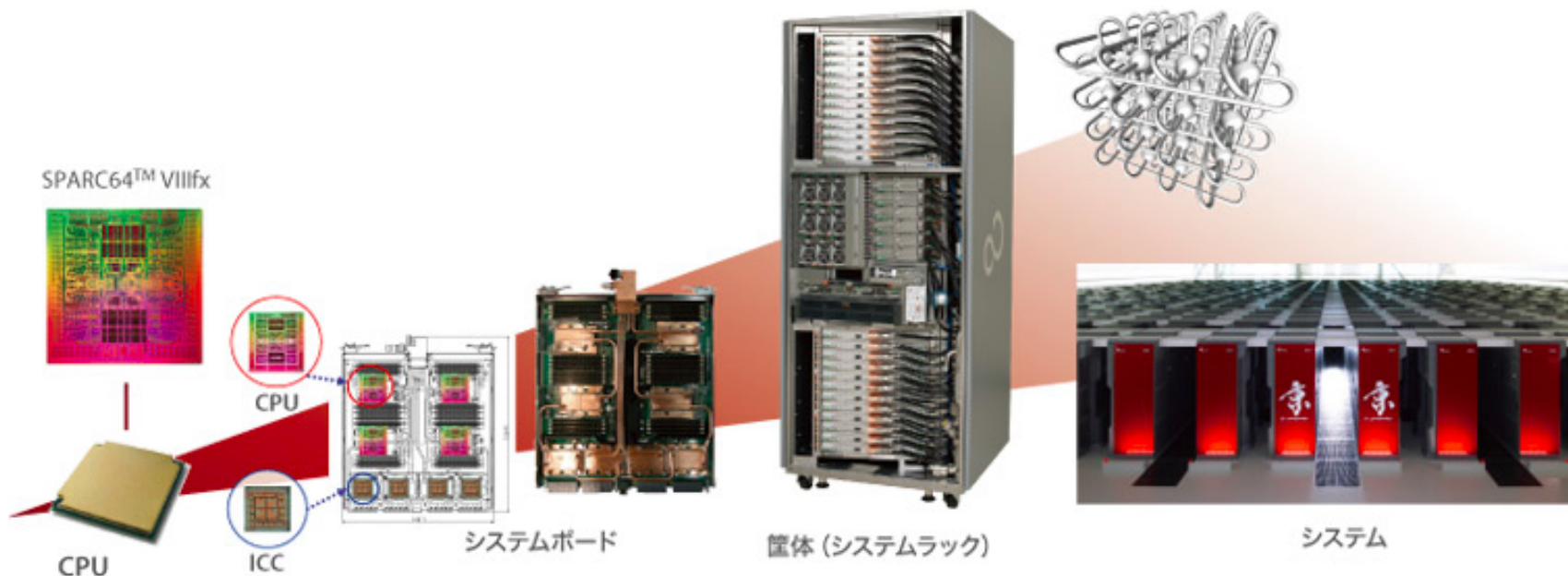
# スーパーコンピュータ

1つのノードに割当て

- コンピュータシミュレーション
  - 物理現象をコンピュータで再現
    - 天気予報・創薬・宇宙の解明
- 膨大な計算を高速に行う
- 研究開発を通じた技術向上へ



空間を細かく格子状に区切って  
雲の量を計算→より詳しい天気予報



参考：京コンピュータ



# スーパーコンピュータの構成

- 特定の計算を高速化(効率化)するため  
特殊な構成が多い
- 自分が持っている汎用コンピュータの構成と比較  
してみよう



計算中に大量のデータを扱いたい  
→メモリ1TB(=1000GB)



低温度(熱)で  
計算したい  
→特殊な油に浸透  
(TSUBAME-KFC)



特定の計算を  
高速化したい  
→GPU8枚





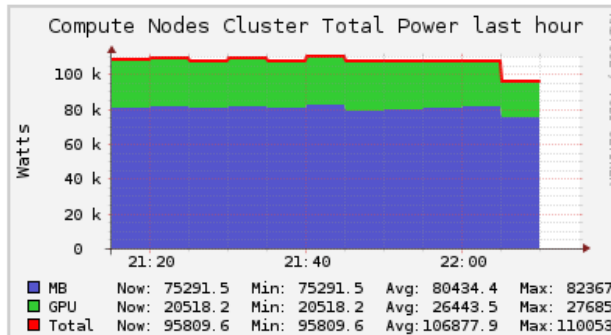
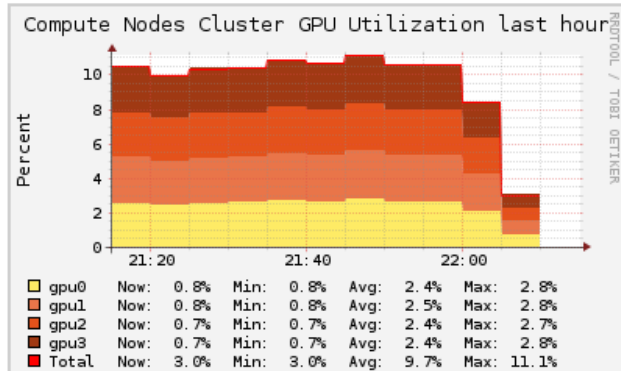
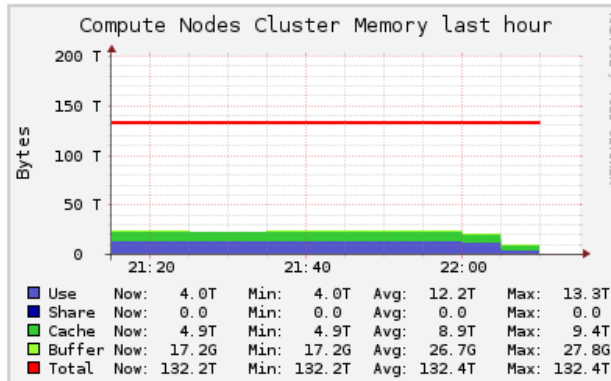
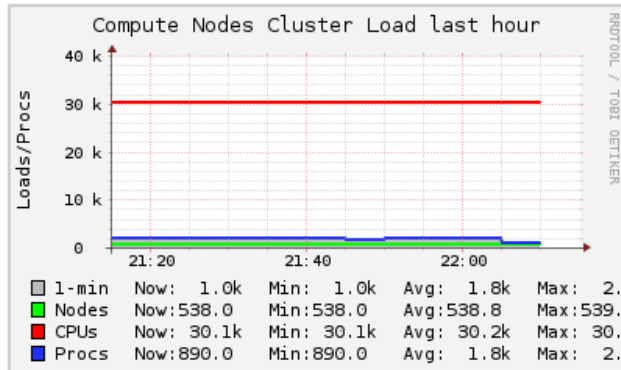
# TSUBAME3.0スーパーコンピュータ

東工大のスーパーコンピュータ

2017年6月のGreen500で世界1位(高電力効率)

倍精度12.15PFLOPS = 12150TFLOPS (地球シミュレータの約300倍)

Overview of Compute Nodes @ 2019-04-09 22:14

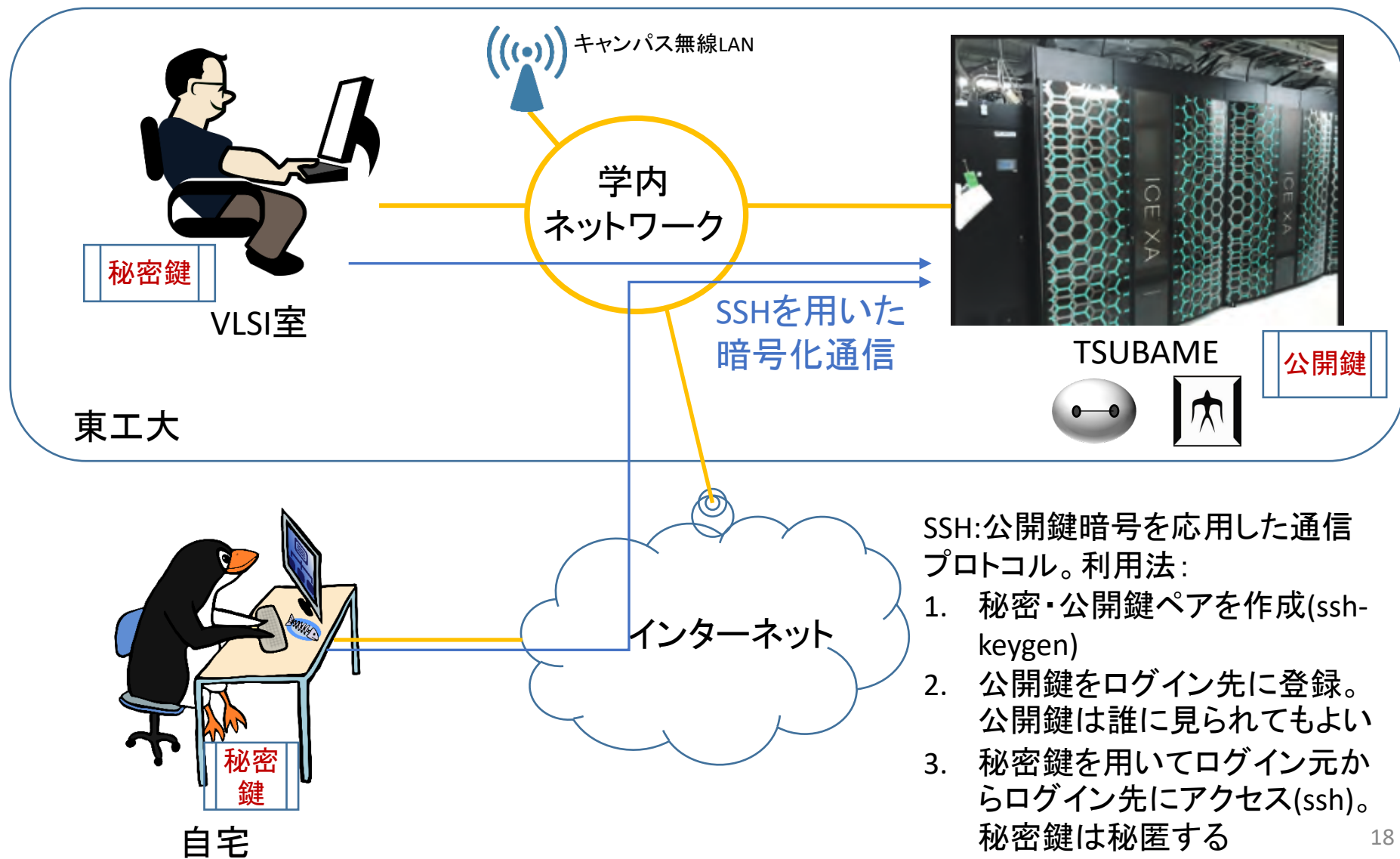


参考:稼働状況

<http://pm1.t3.gsic.titech.ac.jp/ganglia/>



# ネットワークを介した計算機へのアクセス



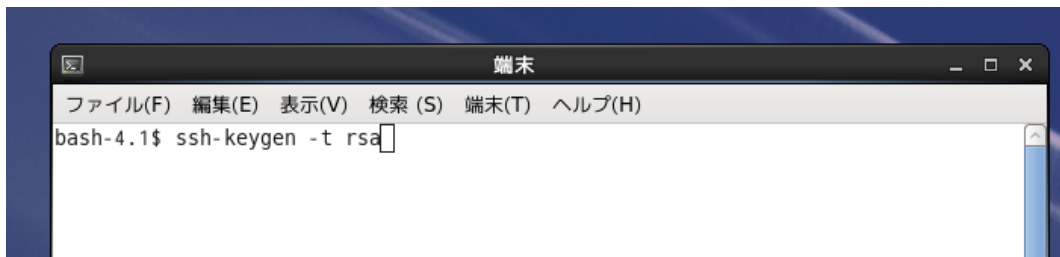
# SSHクライアントソフト

---

- SSHコマンド
  - WindowsのPower Shell内、Mac/Linuxの端末内でデフォルトで利用可
- その他の方法（紹介のみ）
  - PuTTY
    - SSHに対応した端末エミュレータ
    - 対応OS: Win/mac/linux
  - Windows Subsystem for Linux + Ubuntu
    - Windows上でLinuxを実行し、その中からsshコマンドを利用
    - 対応OS: Windows 10

# SSH秘密・公開鍵の生成 (Linuxの場合)

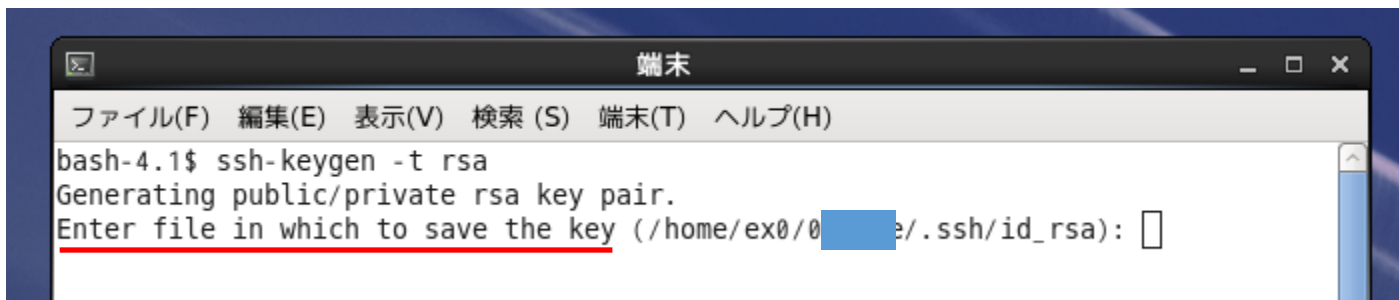
①「端末」にssh-keygen -t rsaと入力、Enterキーを押す



```
bash-4.1$ ssh-keygen -t rsa
```

VLSI室ではLinux  
を用いています

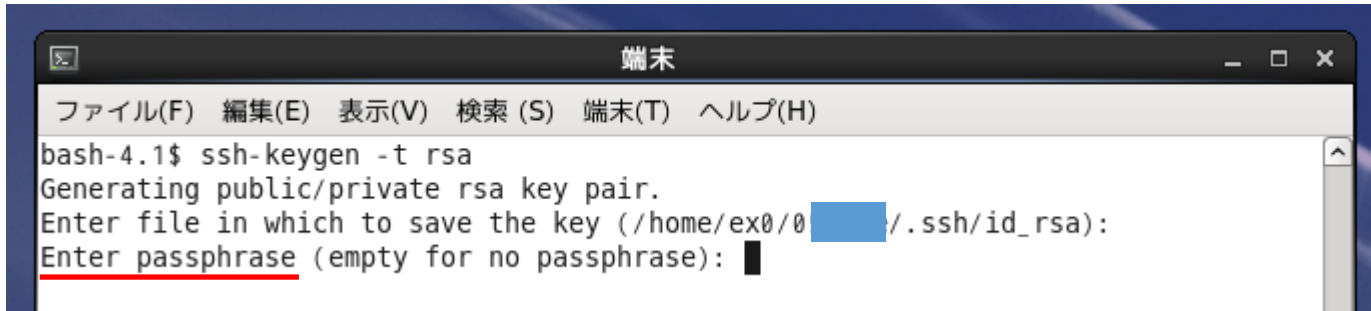
②鍵を保存する場所を入力する。  
デフォルトの場所に保存したいのでEnterキーを押す



```
bash-4.1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ex0/0[redacted]/.ssh/id_rsa):
```

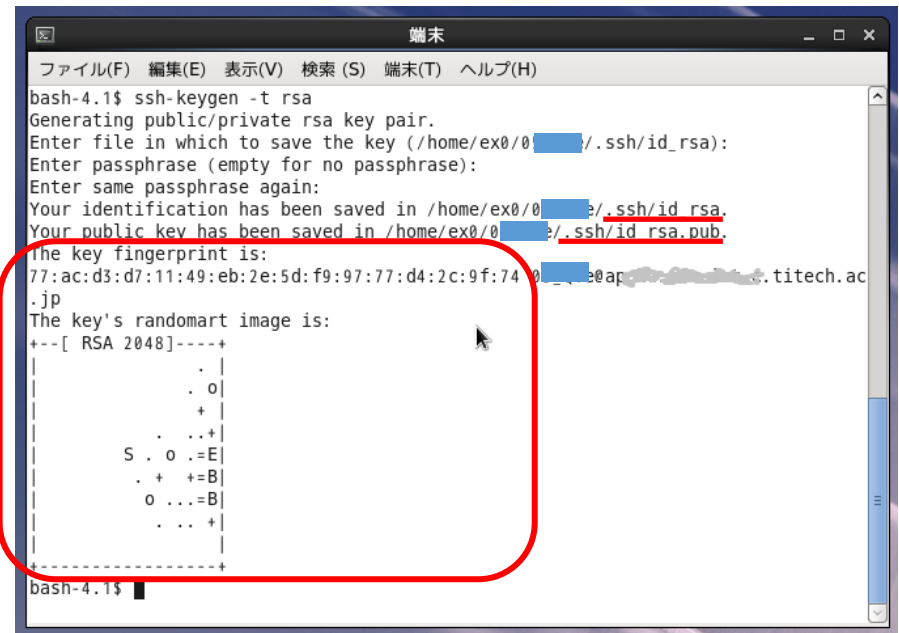
# SSH鍵生成の生成(Linux・つづき)

- ③鍵にアクセスするためのpassphraseを入力する。  
間違いのないように同じpassphraseを2回入力。



```
bash-4.1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ex0/0[redacted]/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
```

- ④Key fingerprintが表示されたら完了  
.sshディレクトリ内に  
id\_rsa(秘密鍵)と  
id\_rsa.pub(公開鍵)



```
bash-4.1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ex0/0[redacted]/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ex0/0[redacted]/.ssh/id_rsa.
Your public key has been saved in /home/ex0/0[redacted]/.ssh/id_rsa.pub.
The key fingerprint is:
77:ac:d3:d7:11:49:eb:2e:5d:f9:97:77:d4:2c:9f:74 [redacted]@ap[redacted].titech.ac.jp
The key's randomart image is:
+--[ RSA 2048 ]-----+
  . |
  . o |
  + |
  . .+ |
  S . o . =E |
  . + +=B |
  o ...=B |
  . . . + |
+-----+
bash-4.1$
```

# SSH秘密・公開鍵の生成 (Windowsの場合)

## POWERSHELLの中で行う

```
Windows PowerShell
PS C:\Users\kunkk> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\kunkk/.ssh/id_rsa):
Created directory 'C:\Users\kunkk/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\kunkk/.ssh/id_rsa.
Your public key has been saved in C:\Users\kunkk/.ssh/id_rsa.pub.
The key's fingerprint is:
SHA256:W3RZWE/q3HT6+Bp7AjDFYztJci2SzFE9nuyj68FEVx8 kunkk@DESKTOP-MI2EFTV
The key's randomart image is:
+--[RSA 2048]--+
|o.=.oEo|
|*Xo+=o|
|.B+Boo=|
|.oo+++.|
|S.o.o+.|
|o o. oo|
|.ooo..|
|...+.|
|.o.o+.|
+--[SHA256]--+
```

① `ssh-keygen -t rsa`  
(keyを生成)

② keyを保存する場所  
(そのままEnterキーを押す)

③ Passphraseを入力  
(入力は表示されない。  
間違いを防ぐため2回聞かれる)

④ keyを保存する場所を開く

秘密鍵

公開鍵(pub)

powershell  
と入力

鍵を保存している場所

```
PS C:\Users\kunkk> cd ./.ssh\
PS C:\Users\kunkk\.ssh> ls
Directory: C:\Users\kunkk\.ssh

Mode                LastWriteTime         Length Name
----                -
-a                4/9/2019   5:05 PM         1675 id_rsa
-a                4/9/2019   5:05 PM          404 id_rsa.pub
```

# TSUBAMEへのSSH公開鍵登録

## TSUBAMEポータルページ

利用者権限：  
アカウント [redacted]  
現在の状態： 利用中  
所属グループ： tga-tslab, tga-egliteracy

[TSUBAME利用状況]  
[ジョブ一覧](#)  
[予約ノード一覧](#)

[利用者情報]  
[利用者情報表示](#)  
[SSH公開鍵登録](#)  
[パスワード設定](#)  
[利用規約一覧](#)

[課金管理]  
[支払コード管理](#)  
[予算コード承認依頼\(0\)](#)

[グループ]  
[所属グループ管理](#)  
[グループ作成](#)  
[履歴表示](#)

### SSH公開鍵追加

SSH公開鍵をコード入力または、ファイルアップロードにて追加することができます。  
SSH公開鍵コードを入力して追加ボタンを押してください。

#### 公開鍵コード入力

公開鍵コードを入力  
又は  
公開鍵ファイルを選択

登録するSSH公開鍵をアップロードしてください。

SSH公開鍵ファイル：  選択されていません

## Tokyo Tech Portal

ログアウト

一般システム

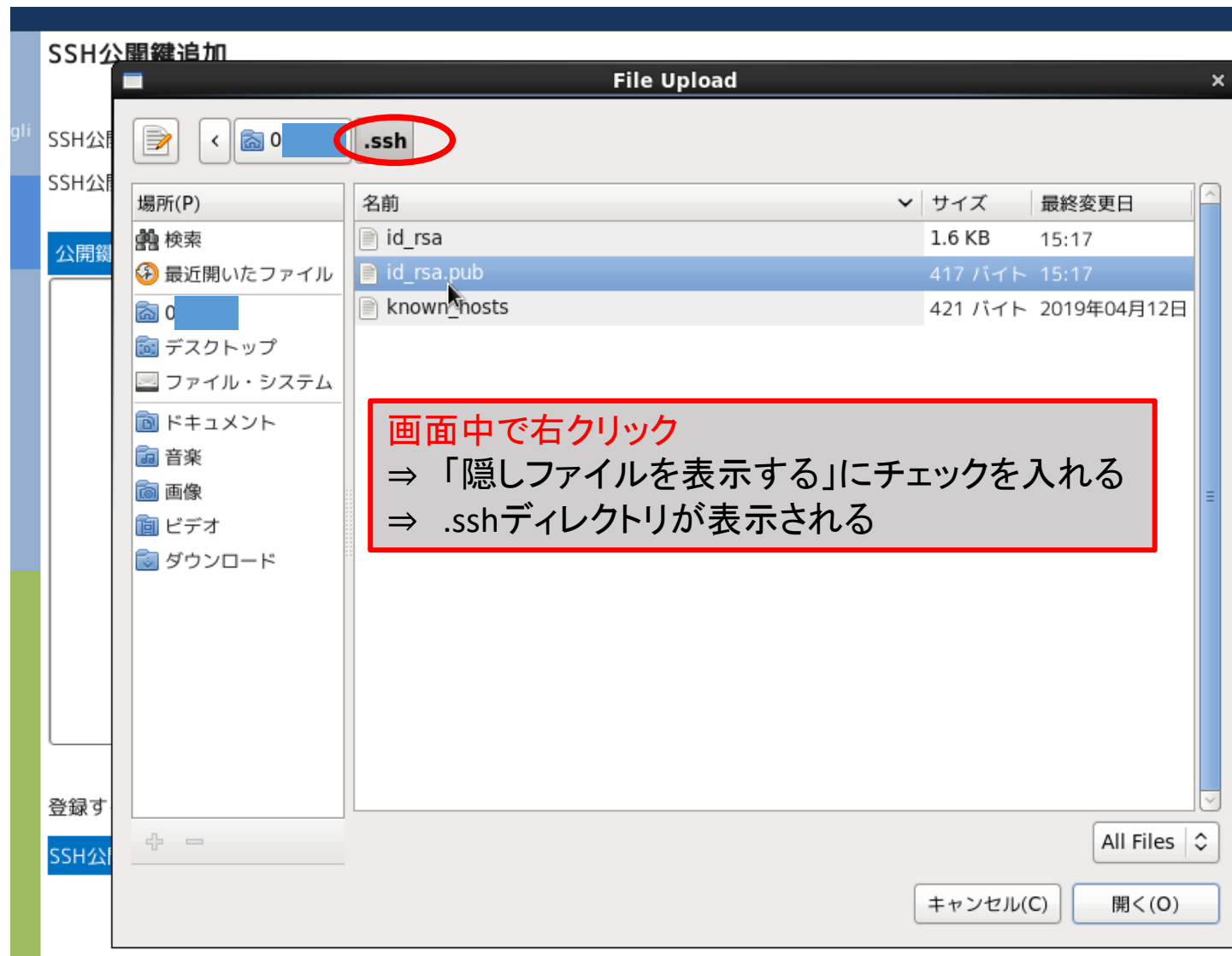
- Tokyo Tech Mail
- 共通メール認証ID
- 学内ネットワークアクセス (SSL-VPN)
- パスワード変更
- 姓名字典登録
- 東工大リサーチリポジトリ(T2R2)
- 図書館サービス: Library Service
- 東工大STARサーチ (STAR Search)
- TSUBAME ポータル**
- T2Report
- 教育用電子計算機システム (学内限定)

① Tokyo Tech Portalに  
アクセス

④ 追加又はアップロード




# 公開鍵(.pub)を選択して開く





# 公開鍵登録完了

利用者権限：  
アカウント：   
現在の状態： 利用中  
所属グループ： tga-tslab,tga-egll  
teracy

[TSUBAME利用状況]  
[ジョブ一覧](#)  
[予約ノード一覧](#)

[利用者情報]  
[利用者情報表示](#)  
[SSH公開鍵登録](#)  
[パスワード設定](#)  
[利用規約一覧](#)

[課金管理]  
[支払コード管理](#)  
[予算コード承認依頼\(0\)](#)

[グループ]  
[所属グループ管理](#)  
[グループ作成](#)  
[履歴表示](#)

## SSH公開鍵追加

SSH公開鍵をコード入力または、ファイルアップロードにて追加することができます。

SSH公開鍵コードを入力して追加ボタンを押してください。

### 公開鍵コード入力

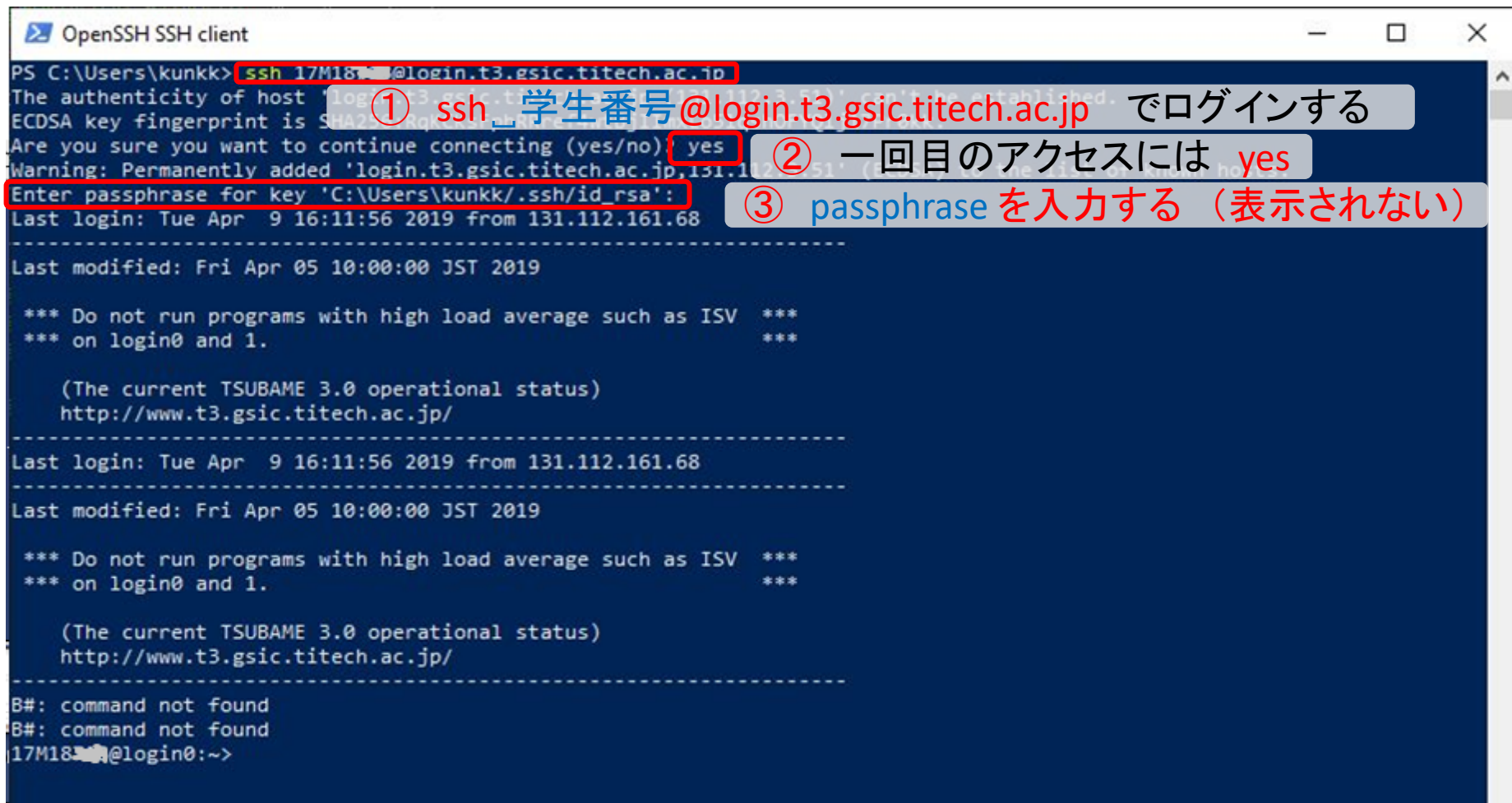
```
ssh-rsa AAAAB3NzaC1yc2EAAAABIJAAAQEA...  
TKJ5xSpPK...  
0NuncR3dKn2MQawdinLUHpDqP...  
e8nSI7Kamsqg...  
...titech.ac.jp
```

登録するSSH公開鍵をアップロードしてください。

SSH公開鍵ファイル： No file selected.

公開鍵ファイル  
を選択したら、  
「アップロード」  
ボタンを押して  
完了

# TSUBAMEへのログイン



```
OpenSSH SSH client
PS C:\Users\kunkk> ssh 17M18...@login.t3.gsic.titech.ac.jp
The authenticity of host 'login.t3.gsic.titech.ac.jp' is not established.
ECDSA key fingerprint is SHA256:Kq...
① ssh_学生番号@login.t3.gsic.titech.ac.jp でログインする
Are you sure you want to continue connecting (yes/no) yes
② 一回目のアクセスには yes
Warning: Permanently added 'login.t3.gsic.titech.ac.jp,131.112.161.68' (ecdsa) to the list of known hosts.
Enter passphrase for key 'C:\Users\kunkk\.ssh\id_rsa':
③ passphraseを入力する (表示されない)
Last login: Tue Apr 9 16:11:56 2019 from 131.112.161.68
-----
Last modified: Fri Apr 05 10:00:00 JST 2019

*** Do not run programs with high load average such as ISV ***
*** on login0 and 1. ***

(The current TSUBAME 3.0 operational status)
http://www.t3.gsic.titech.ac.jp/
-----
Last login: Tue Apr 9 16:11:56 2019 from 131.112.161.68
-----
Last modified: Fri Apr 05 10:00:00 JST 2019

*** Do not run programs with high load average such as ISV ***
*** on login0 and 1. ***

(The current TSUBAME 3.0 operational status)
http://www.t3.gsic.titech.ac.jp/
-----
B#: command not found
B#: command not found
17M18...@login0:~>
```

# ディレクトリ構成と初期設定

- ディレクトリ構成
  - ホームディレクトリ ~/ul>  - TSUBAMEにログインした直後のディレクトリ
  - オプションなしの`cd`コマンドを実行するとこちらに移動
  - 25GBまで利用できます
  - レシピはこのディレクトリの下にダウンロード
- 本講義の共有スペース
  - レシピで使用する学習・評価データ他が置いてある
  - `/gs/hs0/tga-egliteracy`

# コマンド実行

---

```
$ echo _ hello
```

```
hello
```

“\_”はスペース(実際には表示されない)  
注意しながら入力してください

```
$ ls
```

```
ドキュメント ダウンロード デスクトップ
```

```
$ pwd
```

```
/home/6/shinozaki
```

```
$ for _ i _ in _ 1 _ 2 _ 3; do _ echo _ 3*$i=$((3*$i)); done
```

```
3*1=3
```

```
3*2=6
```

```
3*3=9
```

# シェルスクリプト

## 1. テキストファイルを作成

```
#!/bin/bash  
echo hello  
ls  
pwd  
for i in 1 2 3; do echo 3*$i=$((3*$i)); done
```

Bashというシェルスクリプトを使うことを指定

tameshi.sh

## 2. `$ chmod u+x tameshi.sh`

スクリプトを実行できるように設定

## 3. スクリプトを実行

`$ ./tameshi.sh`

先頭に"./"をつけていることに注意

hello

tameshi.sh ドキュメント ダウンロード デスクトップ

/home/6/shinozaki

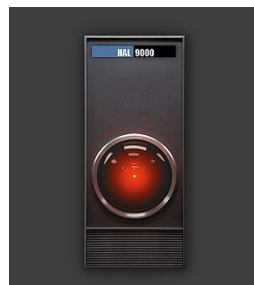
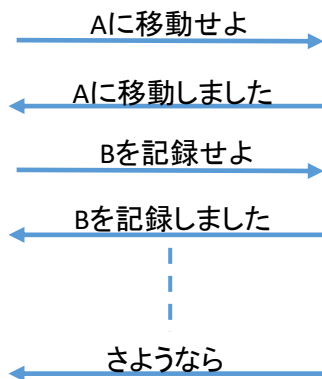
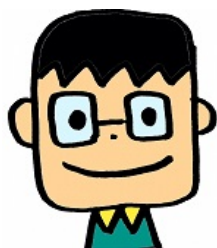
3\*1=3

3\*2=6

3\*3=9

# バッチジョブシステム

## ・対話実行



TSUBAMEでは、大規模な計算はジョブに仕立てて実行する必要がある



## ・バッチジョブ



Aに移動してから、Bを記録する。もしCならDをする。それからEを10回繰り返して、.....。終わったら、「さようなら」と言う

①スクリプトを作成



これお願いします

②スクリプト実行を依頼



引き受けました

③ジョブ管理システムがスクリプトを実行

空いている計算ノードを探したり、混んでいるときに空くのをまったりなどの管理を代行

# ジョブ実行上の注意

- ジョブ実行は課金サービスです



- TSUBAMEポイントの割り当て
  - 一人当たりの持ち分は14400ポイントです。q\_nodeを20時間使用できます
  - 持ち分を越えて使用してはいけません。本講義の目的外に使ってはいけません
  - ポイントのお財布は講義全体で共有しています。無断で持ち分を大きく超えて使用した場合は、減点対象にする場合があります

# TSUBAME課金グループに参加

様

グループtga-egliteracy への招待メールを送信いたします。  
下記のURLをクリックし、グループの参加・不参加の指定を行ってください。

To

This is an invitation email to the group tga-egliteracy. Click the following URL to specify whether or not you will participate in the group.

TSUBAME計算サービス / TSUBAME Computing Services

<http://www.t3.gsic.titech.ac.jp/>

<https://portal.t3.gsic.titech.ac.jp/>

既に全員TSUBAMEアカウント登録を行いグループに参加しているはずですが、もし未だの人は申し出てください



# TSUBAMEポイントの確認

## TSUBAMEポータルページ

利用者権限：  
アカウント：  
現在の状態：利用中  
所属グループ：tga-tslab,tga-egliteracy

### [TSUBAME利用状況]

[ジョブ一覧](#)  
[予約ノード一覧](#)

### [利用者情報]

[利用者情報表示](#)  
[SSH公開鍵登録](#)

[パスワード設定](#)

[利用規約一覧](#)

### [課金管理]

[支払コード管理](#)  
[予算コード承認依頼\(0\)](#)

### [グループ]

[所属グループ管理](#)  
[グループ作成](#)  
[利用報告](#)  
[履歴表示](#)

### 所属グループ一覧

グループ作成年度 2017年度 ~ 2019年度

グループ名

グループ状態 ① ☐ 利用中 ☐ 利用一時停止中 ☐ 利用停止中

### 所属グループ管理

権限 ☐ 一般利用者 ☐ グループ管理者(メイン) ☐ グループ管理者(サブ)

検索

グループ作成

グループ名	コメント	権限	利用状況
		一般利用者	<a href="#">詳細表示</a>
tga-egliteracy	工学リテラシー	一般利用者	<a href="#">詳細表示</a>

②

詳細表示

# TSUBAMEポイントの確認

TSUBAMEポイント欄でチェックできます

TSUBAMEポイント **利用状況確認** 「ポイント購入」ボタンが非アクティブな場合は、上記「利用分野」を設定してください。

所有ポイント	0	ポイント購入
支払コード		支払コード設定

③ 「利用状況確認」押すと  
各自の使用量を確認できます

TSUBAMEグループメンバー毎利用状態

更新

アカウント名	ステータス	2019年度（単位：T3ポイント）														累計
		4月	5月	6月	7月	8月	9月	10月	11月	12月	1月	2月	3月	仮ポイント	年度合計	
	利用中	7,657	0	0	0	0	0	0	0	0	0	0	0	8,640	16,297	16,297
	利用中	432,216	0	0	0	0	0	0	0	0	0	0	0	0	432,216	432,216

# バッチジョブの実行(例)

```
$ cd  
$ cp _/gs/hs0/tga-egliteracy/etc/tameshi.sh _.  
$ qsub _-o _job.log _-e _job.err _tameshi.sh
```

←ドット(.)を忘れないように！

-o \_job.log : ジョブの標準出力を書き出すファイルを指定  
-e \_job.err : ジョブのエラー出力を書き出すファイルを指定  
tameshi.sh : バッチジョブスクリプト

```
#!/bin/bash
```

```
#$ -cwd
```

```
#$ -l q_node=1
```

```
#$ -l h_rt=00:01:00
```

バッチジョブの実行方法  
を指示するオマジナイ

```
echo hello
```

```
ls
```

```
pwd
```

```
for i in 1 2 3; do echo 3*$i=$((3*$i)); done
```

tameshi.sh

# バッチジョブの実行(例)結果確認

---

```
$ cat _job.log
```

hello

foo.err

foo.log

tameshi.sh

/home/6/shinozaki

3\*1=3

3\*2=6

3\*3=9

# 演習課題 (ICT-2)

---

- 別ファイル参照

# 課題提出方法

- 締切: 本日より6日後の朝9時(OCW-iにて指定)
- 提出先: TOKYO TECH OCW-i
- ファイル種別: Text (.txt) file
  - ✓ **NOT** docx, rtf, pdf, png, jpg, etc.
- 投稿タイトル: ICT-2
- フォーマット (**厳格に下記のフォーマットであること**)

```
12B34567, 工大 太郎
Q2.1: No
Q2.2: No
Q2.3: No
```



1行目: 自分の学籍番号と名前

2行目~: 問題ID, ':', 回答(Yes or No)

こちらの答えは(もちろん)ダミー

テンプレートをOCWにアップロードしてあるので、そちらをダウンロードして編集してください

---

# 次回予告

# レシピのダウンロード

---

```
$ cd ~/
```

ダウンロードはホームディレクトリに

レシピのダウンロード(好きなタスクを一つ選んでください)

- 音声認識レシピ

```
$ git clone https://github.com/tttslab/tut-asr-voicecommand.git
```

- 画像認識レシピ

```
$ git clone https://github.com/HirokiNakahara/MNIST_Examples
```

- 機械翻訳レシピ

```
$ git clone https://github.com/kamigaito/tga-egliteracy-seq2seq.git
```



# レシピの構成(各レシピ共通)

runsubame.sh

TSUBAME上でジョブを走らせる準備  
スクリプトrun.shを起動

run.sh

ニューラルネット実験のトップレベルのスクリプト

prep.sh

学習・評価データの準備  
(TSUBAMEでは準備済みなのでスキップ)

train.sh

ニューラルネットの学習

eval.sh

ニューラルネットの評価

# TSUBAMEでのレシピ実行法 (各レシピ共通)

```
$ cd _<directory of the recipe>/exp1p/
```

```
$ qsub _runsubame.sh
```

runsubame.sh : バッチジョブスクリプト

デフォルトは動作確認用の設定 ⇒ GPUを使用して数分程度で終了

## 音声認識の場合の例

```
$ cd _tut-asr-voicecommand/exp1p/
```

```
$ qsub _runsubame.sh
```

# 評価結果の確認方法

---

- 音声認識

```
$ less _ tut-asr-voicecommand/exp1p/score.txt
```

- 画像認識

```
$ less _ MNIST/exp1p/score.txt
```

- 機械翻訳

```
$ less _ seq2seq/exp1p/score.txt
```

# 課金グループの指定(各レシピ共通)

- 10分以上計算時間のかかるジョブ
- ⇒TSUBAMEポイントが必要
  - ✓ 課金グループを指定する必要あり
  - ✓ ジョブの最大実行時間を指定

無駄に長く設定すると無駄にポイントを消費するので注意  
本実習では、どの場合も6時間あれば足りる

```
$ qsub -g tga-egliteracy -l h_rt=06:00:00 runsubame.sh
```

-g tga-egliteracy : 課金グループ(お財布)指定  
-l h\_rt=06:00:00 : ジョブの最大実行時間 (hh:mm:ss)指定  
-p -4 : ジョブの優先度指定(すこし高めでポイントを多く消費)  
runsubame.sh : バッチジョブスクリプト

## 音声認識の場合の例

```
$ cd tut-asr-voicecommand/exp1p/
```

```
$ qsub -g tga-egliteracy -l h_rt=06:00:00 runsubame.sh
```

# (参考) Google Colabでの実行法

GPUが無料で最大12時間まで連続使用できます

画像認識レシピの場合

1. Googleドライブにアクセス(アカウントがなければ作る)
2. Googleドライブにcolab.ipynbファイルをアップロード
3. colab.ipynbを右クリック→その他→Colaboratory で起動  
(Colaboratoryがなければ“アプリを追加”から追加する)
4. Shift+Enterで各セルが実行できる(Ctrl+F9で全部実行)

実行例:

```
[1] # Check GPU status
!nvidia-smi
```

📄 Tue Apr 9 04:00:21 2019

```
+-----+
| NVIDIA-SMI 418.56      Driver Version: 410.79      CUDA Version: 10.0      |
+-----+-----+-----+-----+
| GPU  Name      Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+
|   0   Tesla K80           Off | 00000000:00:04:0 Off |                    0 |
| N/A   29C    P8      29W / 149W |  0MiB / 11441MiB |      0%      Default |
+-----+-----+-----+-----+
```