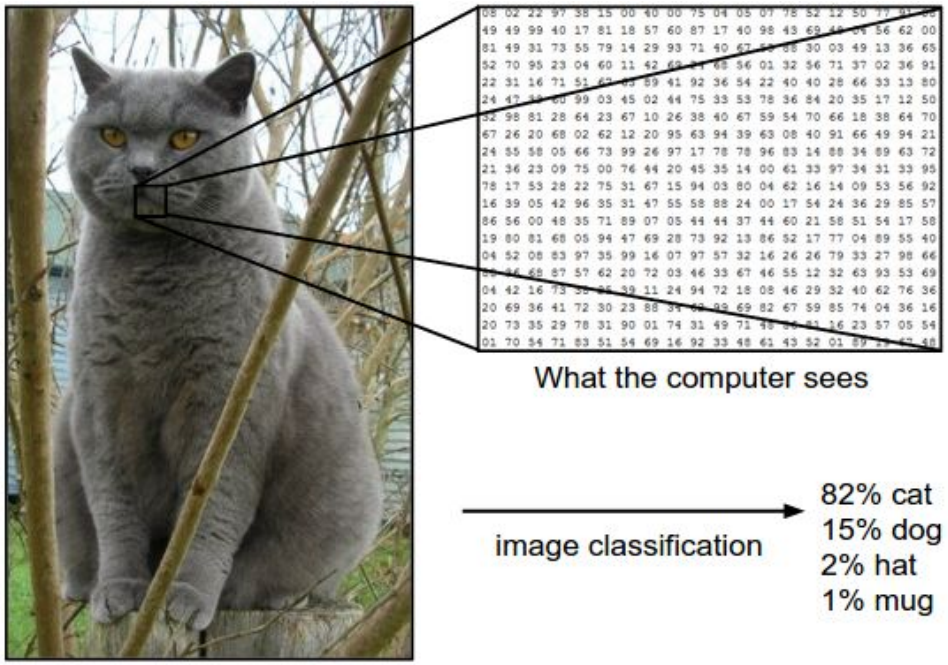
Verb Detection
Group B

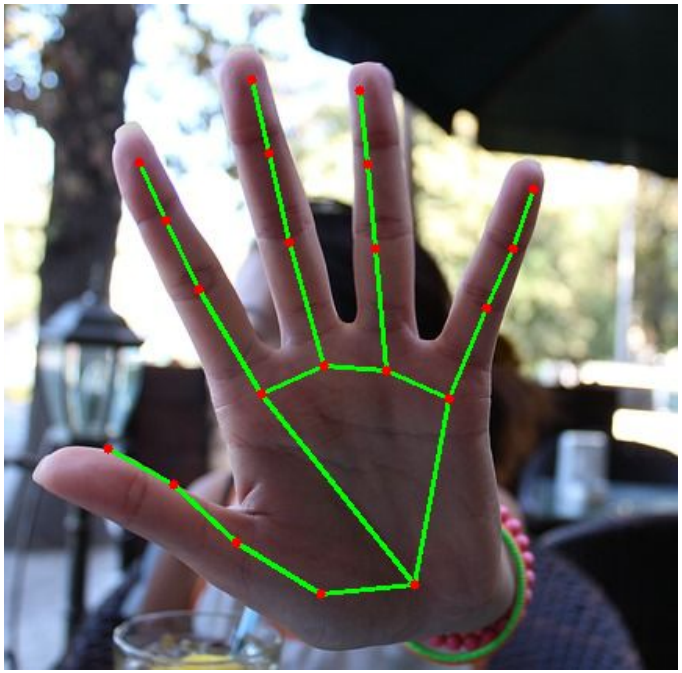
Siyi Dai

Motivation

- **Known:**
  - Objects + Positions
  - Noun (object interacting with hands)
- **Goal:**
  - Detect **Action/Verb**
- **Requirement:**
  - **Accuracy** of verb detection
  - **Runtime speed** for real-time detection

Motivation





Classic Image Classification:
256x256x3 or even more!

Hand Landmarks Classification:
only **21** points per hand!

I. Lu, Dengsheng, and Qihao Weng. "A survey of image classification methods and techniques for improving classification performance."
II. Zhang, Fan, et al. "Mediapipe hands: On-device real-time hand tracking."

Tasks

Dataset Preparation

- Data Recording and Extraction
- Dataset Generation

LSTM Model

- Data Pre-process
- Data Loader
- Model Building
- Model Training

Results Testing

- Verb Pipeline

Dataset Preparation

- **Data Recording and Extraction**
  - Data **Recording**
    - **8** rosbags, **4** subjects, **Ego** perspective
    - Duration: **3-4mins**
      - **5** rosbags, **3** subjects → **training**
      - **3** rosbags, **3** subjects → **testing**
  - Data **Extraction**
    - Republish compressed images
    - Image saver
    - save in *.jpg

Dataset Preparation

- **Dataset Generation**
  - **Label** the **images** from each bags into **12** classes
    - "close": 0,
    - "decorate": 1,
    - "flip": 2,
    - "move_object": 3,
    - "open": 4,
    - "pick_up": 5,
    - "pour": 6,
    - "put_down": 7,
    - "screw": 8,
    - "shovel": 9,
    - "squeeze": 10,
    - "other": 11

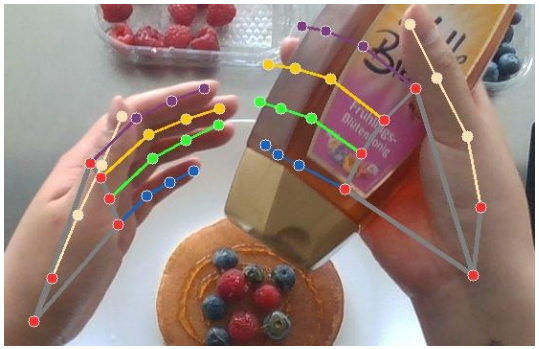
Consistency in Criteria →
Label all of the images:
37805 images

Dataset Preparation

- **Dataset Generation Tool**
  - **Folder** Structure for each recording/rosbag

```
sy0
├── close
│   └── sy0_0
│       ├── images
│       └── close_sy0_0.csv
```

    - Level 0: Recording name
    - Level 1: Verb name
    - Level 2: Recording_Sequence
    - Level 3: Image
  - Generate hand landmarks with **Mediapipe**
    - Detecte 21 points per hand
    - landmark_pb2.NormalizedLandmarkList
    - Output as DataFrame
    - Save in .json/**.csv**



LSTM Model

- **Data Pre-process**
  - Drawbacks of Mediapipe: **no detection, no output**
  - **High** requirement in recordings! → we re-recorded for 3 times.

- Proper move speed
- Steady perspective
- Show complete hand
- No other subjects
- (show complete objects)

Figure 18: The “Perfectly Wrong” Example (speed x 0.5)

LSTM Model

- **Data Pre-process**
  - Drawbacks of mediapipe: **no detection, no output**
    - **Complete features:**
      - **zero-padding** when there's only one hand
    - **Continuous input:**
      - **zero-padding** when there's no detection at all
  - The number of features: **126 = 21 points x (x, y, z) x (left, right)**

Figure 19: The No Detection Examples

LSTM Model

- **Data Loader**
  - Temporal sequence → LSTM
    - Proper time steps and a stride for sliding window

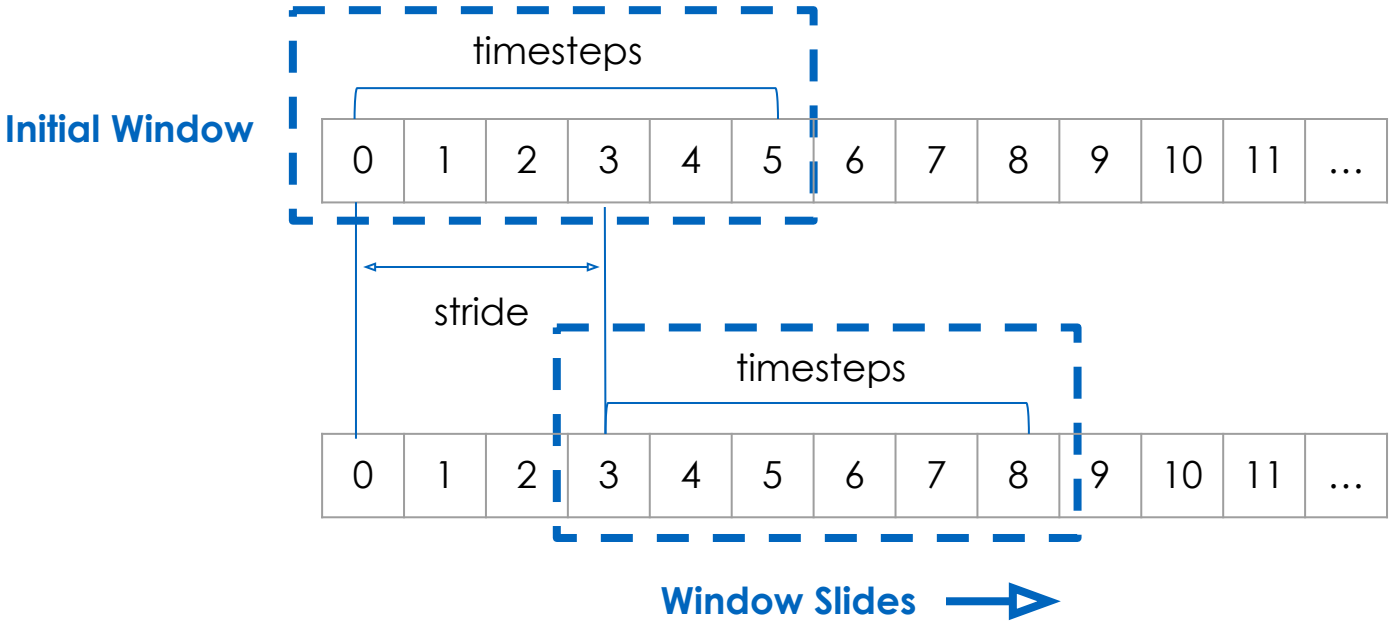


Figure 20: Sliding window demo

I. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory."

LSTM Model

Runtime =
Preprocess time + Predict time

- **Data Loader Parameters Tuning**

(Stride, Time_steps)	Accuracy ↑	Macro Avg f1-score ↑	Number of Classes Accuracy >= 0.85 ↑	Runtime (s) ↓	
				Mean ↓	Std ↓
(10, 100)	0.85	0.68	2/12	7.531	0.025
(5, 50)	0.78	0.60	1/12	3.981	0.014
(5, 20)	0.74	0.47	1/12	1.830	0.013
(1, 20)	**0.98**	0.93	11/12	1.837	0.020
(1, 10)	**0.98**	**0.94**	**12/12**	0.785	0.009
(1, 5)	0.97	0.89	8/12	**0.524**	**0.006**

Table 1: Stride and Timesteps Combination Comparison under 2 layers of LSTM with 128 units and 64 units. Execution time is calculated by 100 iterations with Google Colab GPU accelerator

LSTM Model

- **Model Building**

Model	Accuracy ↑	Macro Avg f1-score ↑	Number of Classes Accuracy >= 0.85 ↑	Inference Time (ms) ↓	
				Mean ↓	Std ↓
1 x LSTM, 64	0.96	0.88	8/12	4.837	0.752
1 x LSTM, 128	0.97	0.91	11/12	**4.180**	**0.341**
Encoder with 2 x LSTM, 128, 64	**0.98**	**0.94**	**12/12**	4.922	0.489
Autoencoder with 2 x LSTM, 128, 64 2 x LSTM, 64, 128	0.97	0.90	9/12	6.323	0.552

Table 2: Model Comparison with stride = 1 and timesteps = 10,
Inference time is calculated by 1000 iterations with Google Colab GPU accelerator

LSTM Model

- **Model Building**

Model: "**LSTM_128**"

Layer (type)	Output Shape	Param #
0_LSTM (LSTM)	(None, 10, **128**)	130560
1_Dropout (Dropout)	(None, 10, 128)	0
2_Flatten (Flatten)	(None, 1280)	0
3_Dense (Dense)	(None, 128)	163968
4_Dense (Dense)	(None, 12)	1548

Total params: 296,076
Trainable params: 296,076
Non-trainable params: 0

Table 3: Model Summary

I. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory."

LSTM Model - Classification Report

	precision	recall	f1-score	support
close	0.91	0.87	0.89	60
decorate	0.96	0.95	0.96	288
flip	0.87	0.89	0.88	80
move_object	0.96	0.84	0.90	76
open	0.82	0.97	0.89	80
pick_up	0.94	0.94	0.94	431
pour	0.98	0.95	0.96	165
put_down	0.89	0.85	0.87	209
screw	0.96	0.95	0.95	333
shovel	0.87	0.77	0.81	209
squeeze	0.91	0.96	0.93	171
other	0.99	0.99	0.99	5423
accuracy			0.97	7525
macro avg	0.92	0.91	0.91	7525
weighted avg	0.97	0.97	0.97	7525

Table 4: Classification Report

```
"patience": 8,
"validation_split": 0.3,
"epochs": 100,
"batch_size": 32,
```

Results Testing - Verb pipeline

- **Data Pre-process**
  - Zero-paddings → **Content-fillings**
    - 6000 frames → small batch
      - What if the timestamp gap is not in the middle, but in the beginning or the end?
      - What if in this batch, there is no detection at all?
      - What if …?
  - The number of features: **126 = 21 points x (x, y, z) x (left, right)**
  - Data Window: **2D → 3D**
  - **Timestamp** Check
- **Visualization** for Hand Landmarks
  - Debug and visualizer
- Model **Load** and **Predict**

Results Testing - Verb pipeline

- **Runtime Optimization**
  - Load model while predict → integrated pipeline **2788.24ms**
  - Load model before predict → much faster
    - predict → 37.22ms
    - **predict_on_batch → 4.18 ms**
  - Load labels before predict
- **Runtime = Preprocess time + Predict time**
  - → mean = **804.66ms**, std = 8.31ms
    - calculated by 500 iterations with Google Colab GPU accelerator
  - **Original: 2788.24ms → Final: 804.66ms**
- **Runtime reduction: 71% ↓**

Results Testing - Verb pipeline

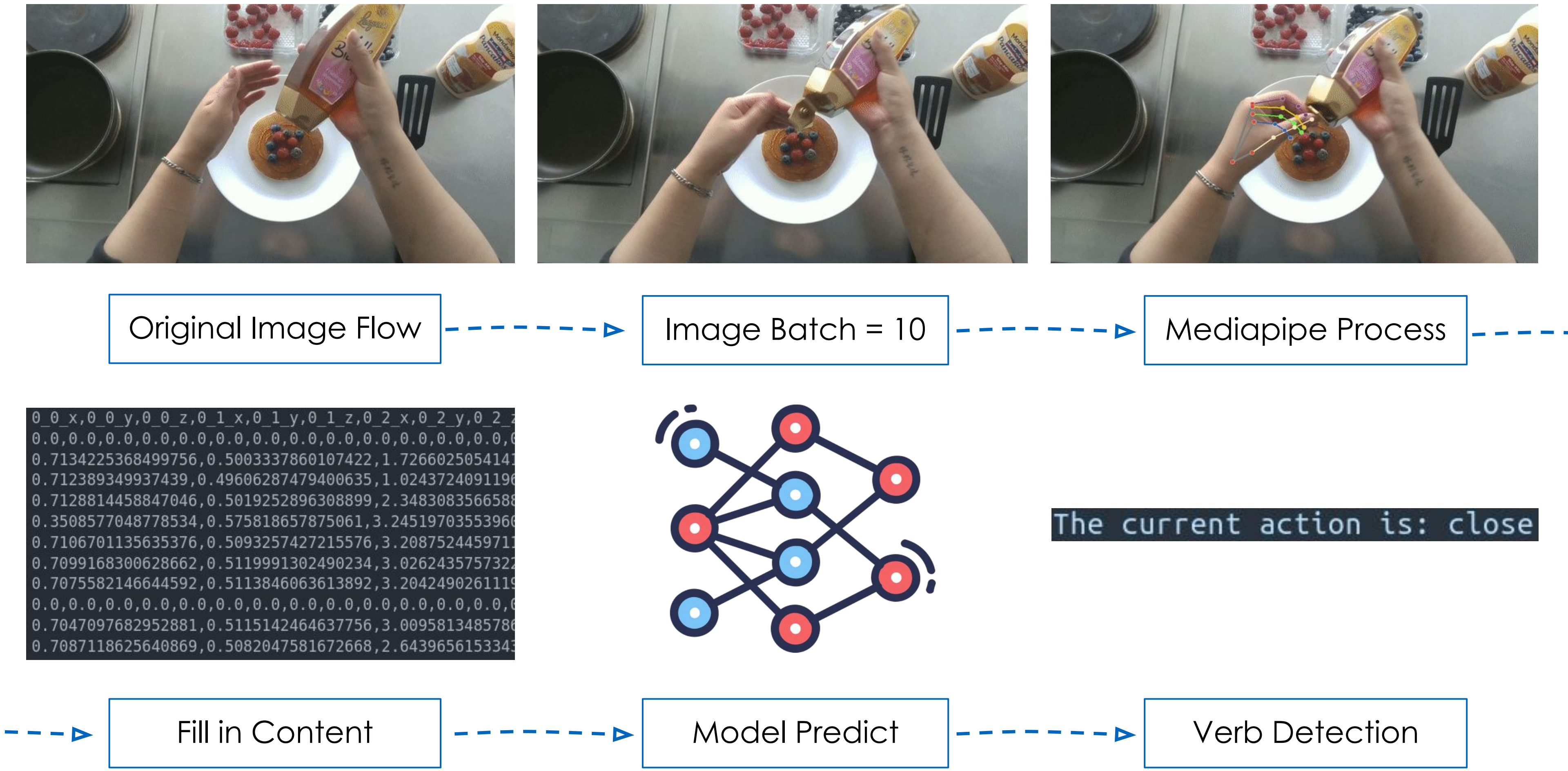


Figure 20: Verb Pipeline Workflow