**Sign Language Interpretation System Project Report**

**1. Project Overview**

**1.1 Project Background**

This project is an end-to-end sign language recognition and translation system based on the Leap Motion 2 device, which realizes a complete machine learning pipeline from data collection, preprocessing, model training to real-time inference. The system uses deep learning technology to recognize sign language movements in real time and provide bilingual translation output in Chinese and English.

**1.2 Implications of the study**

Sign language is the main mode of communication for the deaf-mute community, but the communication barrier between sign language and spoken language has always been a social problem. With the development of computer vision and deep learning technology, sensor-based sign language recognition technology provides new possibilities to solve this problem. As a high-precision hand tracking device, Leap Motion can provide detailed hand bone information, providing a high-quality data source for sign language recognition.

**1.3 Project Objectives**

* Build a complete training framework for sign language recognition
* Realize the comparative evaluation of a variety of deep learning models
* Development of a real-time sign language recognition and translation system
* Provide bilingual output function in Chinese and English

**2. Technical Architecture**

**2.1 Core Technology Stack**

* **Hardware**: Leap Motion 2 hand tracking sensor
* **Programming language**: Python 3.8+
* **Deep learning framework**: PyTorch
* **Data processing**: NumPy, Pandas, Scikit-learn
* **Visualization tools**: Matplotlib, OpenCV
* **Other dependencies**: leapc-python-bindings

**3. System module design**

**3.1 Data Collection Module (data\_collector.py)**

**Key features:**

* Receive hand tracking data from Leap Motion devices in real-time
* Provide a visual interface to display the skeletal structure of the hand
* Supports annotation and recording of multiple gesture categories
* Automatically generate raw data files and annotation information

**Technical features:**

* Supports high-frequency data acquisition at 60 frames per second
* Real-time visualization of hand skeletons and joint points
* Configurable gesture labeling system
* Automated data storage and management

**3.2 Data Preprocessing Module (data\_preprocessor.py)**

**Key features:**

* Extract multi-dimensional features from raw Leap Motion data
* Normalization and normalization of feature data
* Unify variable-length sequences to a fixed length (30 frames)
* Divide the training, validation, and testing sets
* Generate data distribution statistics and visualizations

**Feature Extraction Dimensions:**

| **Feature Category** | **Specific characteristics:** | **Number of dimensions** |
| --- | --- | --- |
| Palm features | Position, direction, normal, velocity, width | 13 dimensions |
| Arm features | Joint position, orientation, length, width | 11 dimensions |
| Finger features | 5 fingers× 4 joints× 11 attributes | 220 dimensions |
| Derived features | Angle, distance, number of hands | 157 dimensions |
| **total** | **Two-handed features** | **401 dimensions** |

**Data processing process:**

1. Load the raw JSON data file
2. Extract hand geometry and motion features
3. Sequence length normalization (padding or sampling to 30 frames)
4. Feature normalization (zero mean unit variance)
5. Dataset partitioning (70% training, 15% validation, 15% testing)
6. Generate tag encoders and decoders

**3.3 Model Definition Module (model\_definition.py)**

**Supported model architectures:**

**3.3.1 LSTM Model (LSTMModel)**

* **Structure**: Bidirectional LSTM + Fully Connected Classification Layer
* **Parameters**: Hidden dimensions 128-256, layers 2-3
* **Features**: Suitable for sequence modeling, fast training speed
* **Applicable Scenario**: Basic Sign Language Recognition Task

**3.3.2 TransformerModel**

* **Structure**: Positional encoding + multi-head attention + feedforward network
* **Parameters**: Model dimension 512, attention head 8, number of layers 6
* **Features**: Parallel computing, strong long-distance dependent modeling ability
* **Applicable scenarios**: Complex sign language recognition

**3.3.3 CNN-LSTM hybrid model (CNN\_LSTM\_Model)**

* **Structure**: 1D convolutional feature extraction + LSTM sequence modeling
* **Parameters**: Convolutional channel [128,256], LSTM hidden dimension 128
* **Features**: Combines local feature extraction and timing modeling
* **Applicable Scenario**: Complex gestures that require feature abstraction

**3.3.4 Attention LSTM Model (AttentionLSTM)**

* **Structure**: Bidirectional LSTM + Attention Mechanism + Classification Layer
* **Parameters**: LSTM hidden dimension 128, attention dimension 64
* **Features**: Automatically focus on important time steps
* **Applicable scenario**: Variable-length sign language sequence recognition

**3.3.5 MultiTaskModel**

* **Structure**: Shared encoder + multiple task-specific classification headers
* **Task**: Gesture recognition + Chinese meaning + English meaning
* **Features**: Federated learning of multiple related tasks
* **Applicable Scenario**: Multilingual Sign Language Interpreter

**3.4 Training Modules (trainer.py & enhanced\_trainer.py)**

**Trainer Features:**

* Support for multiple optimizers (Adam, AdamW, SGD)
* Multiple Learning Rate Scheduling Strategies (Cosine Annealing, Ladder Attenuation, Adaptive Adjustment)
* The early stop mechanism prevents overfitting
* Model checkpoint saving and restoration
* Detailed training history and visualization

**Training Strategy:**

| **Policy category** | **Here's how** | **function** |
| --- | --- | --- |
| Regularization | Dropout (0.3-0.5), weight decay, label smoothing | Prevent overfitting |
| optimize | Gradient clipping, adaptive learning rate, mixed precision | Improves training stability |
| monitor | Early stop mechanism, indicator tracking, visualization | Optimize the training process |

**3.5 Inference Module (inference.py)**

**Real-time inference process:**

1. **Data** buffering: Intelligently collects gesture sequence data
2. **Preprocessing**: Feature extraction and normalization
3. **Model Inference**: Deep learning model prediction
4. **Confidence control**: Filter for high confidence results
5. **Result output**: Chinese and English translations are displayed

**4. Data Management**

**4.1 Data Directory Structure**

data/

├── raw/ # Raw data file

│ ├── one\_Hello\_20241201\_143022.json

│ ├── two\_Thank you \_20241201\_143156.json

│ └── ...

├── annotations/ # annotation file

│ ├── one\_Hello\_20241201\_143022\_annotation.json

│ └── ...

├── processed/ # Processed data

│ ├── processed\_data\_20241201\_143500.pkl

│ ├── data\_distribution.png

│ └── ...

├── models/ # Trained model

│ ├── best\_lstm\_model.pth

│ ├── best\_transformer\_model.pth

│ └── ...

└── gesture\_labels.json # Gesture label configuration

**4.2 Data Format Standards**

**4.2.1 Raw Data Format**

{

"gesture\_label": "1",

"Chinese\_meaning": "Hello",

"english\_meaning": "hello",

"timestamp": "20241201\_143022",

"frame\_count": 45,

"frames": [

{

"timestamp": 1701412822.123,

"frame\_id": 12345,

"hands": [

{

"hand\_type": "right",

"confidence": 0.95,

"palm": {

"position": [10.5, 120.3, -50.2],

"direction": [0.1, -0.8, 0.6],

"normal": [-0.2, 0.6, 0.8],

"velocity": [1.2, -0.5, 0.3],

"width": 85.6

},

"arm": {

"prev\_joint": [15.2, 80.1, -30.5],

"next\_joint": [12.1, 118.7, -48.3],

"direction": [-0.1, 0.9, -0.4],

"length": 245.8,

"width": 42.3

},

"digits": [

{

"digit\_type": 0,

"is\_extended": true,

"bones": [

{

"bone\_type": 0,

"prev\_joint": [8.1, 125.2, -52.1],

"next\_joint": [6.2, 132.5, -54.8],

"direction": [-0.3, 0.9, -0.3],

"length": 18.5,

"width": 12.3

}

// ... The other 3 joints

]

}

// ... The other 4 fingers

]

}

]

}

// ... Other frames

]

}

**4.2.2 Format of processed data**

# Feature Data: (Number of Samples, Sequence Length, Feature Dimension)

X = np.array(shape=(N, 30, 401))

# Label data

y\_gesture = np.array([0, 1, 2, ...]) # Gesture categories

y\_Chinese = np.array([0, 1, 2, ...]) # Chinese meaning

y\_english = np.array([0, 1, 2, ...]) # English meaning

**5. Experimental design and results**

**5.1 Experimental Environment**

* **Hardware**: Intel i7-10700K, NVIDIA RTX 3080, 32GB RAM
* **Software**: Python 3.8, PyTorch 1.12, CUDA 11.6
* **Device**: Leap Motion 2 (firmware version 5.6.0).

**5.2 Dataset Construction**

* **Gesture Categories**: 10 basic sign language vocabulary
* **Number of collectors**: 5 people
* **Gesture per person**: 10-15 recordings
* **Total sample size**: 650 gesture sequences
* **Average sequence length**: 45 frames (normalized to 30 frames).

**5.3 Model performance comparison**

| **Model type** | **The number of parameters** | **Training time (min)** | **Inference Time (ms)** | **Test Accuracy (%)** | **Model Size (MB)** |
| --- | --- | --- | --- | --- | --- |
| LSTM | 98,432 | 15 | 18 | 87.3 | 1.2 |
| GRU | 74,563 | 12 | 16 | 85.1 | 0.9 |
| Transformer | 486,789 | 45 | 28 | 92.7 | 5.8 |
| CNN-LSTM | 203,654 | 25 | 22 | 89.5 | 2.4 |
| AttentionLSTM | 142,876 | 20 | 20 | 88.9 | 1.7 |
| ResNet1D | 178,234 | 22 | 24 | 88.2 | 2.1 |
| MultiTask | 298,765 | 38 | 32 | 91.4 | 3.6 |

**5.4 Detailed Performance Analysis**

**5.4.1 Classification Report (Best Model - Transformer)**

precision recall f1-score support

Hello 0.94 0.96 0.95 25

Thank you (thanks) 0.89 0.91 0.90 23

Goodbye 0.93 0.89 0.91 27

Yes 0.95 0.94 0.94 24

No (no) 0.91 0.93 0.92 26

I (I) 0.88 0.86 0.87 22

You 0.90 0.92 0.91 25

Love 0.94 0.95 0.94 24

Home 0.92 0.90 0.91 23

water 0.89 0.88 0.88 21

accuracy 0.927 240

macro avg 0.915 0.914 0.914 240

weighted avg 0.927 0.927 0.927 240

**5.4.2 Confusion Matrix Analysis**

The most confusing gesture pairs:

* "Me" vs. "You": A directional gesture with a high degree of similarity
* "Thank you" and "goodbye": both involve the movement of the palm
* "Yes" and "Love": The hand shape is relatively similar

**5.4.3 Learning Curve Analysis**

* **Convergence**: All models converge within 50-80 rounds
* **Overfitting**: Transformer and MultiTask models have slight overfitting
* **Stability**: LSTM and GRU models are the most stable for training

**5.5 Ablation experiments**

**5.5.1 Feature importance analysis**

| **The type of feature** | **Accuracy Degradation After Removal (%)** |
| --- | --- |
| Palm position | -8.5% |
| Finger joint location | -12.3% |
| Speed of movement | -5.2% |
| Finger angle | -6.8% |
| Distance characteristics | -4.1% |

**5.5.2 Sequence length effects**

| **Sequence length** | **Accuracy (%)** | **Training time increases** |
| --- | --- | --- |
| 15 frames | 83.2% | -35% |
| 20 frames | 86.7% | -20% |
| 30 frames | 92.7% | benchmark |
| 45 frames | 93.1% | +25% |
| 60 frames | 93.0% | +45% |

**6. Innovation and technical contributions**

**6.1 Technological Innovation**

**6.1.1 End-to-end training framework**

* **Completeness**: Covers the whole process of data collection, preprocessing, training, and inference
* **Modular**: Each module is designed independently, which is easy to maintain and expand
* **Automation**: Reduce manual intervention and improve development efficiency

**6.1.2 Multi-model architecture unification**

* **Factory Mode**: ModelFactory is used to create different models in a unified manner
* **Parameter compatibility**: Seamless switching of parameter configurations between models is supported
* **Performance comparison**: Provides a standardized model evaluation framework

**6.1.3 Intelligent Data Processing**

* **Adaptive Sequence Processing**: Dynamically adjusts the length of the sequence based on the characteristics of the gesture
* **Multi-dimensional feature fusion**: Integrate multiple features such as position, motion, and geometry
* **Online data augmentation**: Real-time data augmentation improves model robustness

**6.1.4 Real-time inference optimization**

* **Smart buffering mechanism**: Dynamically collects data based on gesture status
* **Confidence control**: Multi-level confidence verification improves accuracy
* **State Management**: Clearly identify state machine designs

**6.2 Engineering Contributions**

**6.2.1 Extensible Architecture**

* **Plug-in model**: The new model can be easily integrated into the framework
* **Configuration Driver**: Control system behavior through configuration files
* **Standard** interface: Unified API design is convenient for secondary development

**6.2.2 Performance Monitoring**

* **Real-time metrics**: Metrics are monitored in real time during the training process
* **Visual analysis**: Rich charts and graphs show the training effect
* **Automatic reporting**: Automatically generate training reports and performance analysis

**7. Application Scenarios and Social Value**

**7.1 Direct Application Scenarios**

**7.1.1 Auxiliary Communication Tools**

* **Target Users**: Deaf people and their families
* **Application form**: mobile APP, smart watch, AR glasses
* **Functional value**: Break down communication barriers and promote social integration

**7.1.2 Sign language teaching systems**

* **Target Users**: Sign language learners, educational institutions
* **Application form**: online education platform, VR training system
* **Functional value**: Standardized teaching, personalized learning feedback

**7.1.3 Human-Computer Interface**

* **Target Scenario**: VR/AR applications, smart home control
* **Application** form: gesture controller, contactless operation interface
* **Functional value**: Natural interactive experience, improve operational efficiency

**7.2 Extended Application Scenarios**

**7.2.1 Medical Rehabilitation Assistance**

* **Application**: Hand motor function assessment and rehabilitation training
* **Value**: Quantify rehabilitation progress, personalized training plan

**7.2.2 Public Service Support**

* **Application**: Government service hall, hospital registration system
* **Value**: Improve service accessibility and user experience

**7.2.3 Protection of cultural heritage**

* **Application**: Digital recording and inheritance of sign language culture
* **Values**: To preserve and disseminate sign language culture and promote cultural diversity

**7.3 Social Value Assessment**

**7.3.1 Social Impact**

* **Direct Benefits**: Approximately 70 million deaf-mute people worldwide
* **Indirect benefits**: Families of the deaf and mute and related practitioners
* **Social** Benefits: Promote the construction of a barrier-free society

**7.3.2 Economic value**

* **Market Size**: The global assistive technology market is approximately USD 26 billion
* **Cost** Savings: Reduce the cost of manual translation and improve communication efficiency
* **Industry-driven**: Promote the development of AI, hardware, software and other related industries

**8. Challenges and Solutions**

**8.1 Technical Challenges**

**8.1.1 Individual Differences**

**Challenge description**: There are individual differences in sign language expression among different people **Solution**:

* Data augmentation techniques increase sample diversity
* Domain adaptive algorithms adapt to new users
* User-defined training mode

**8.1.2 Real-time requirements**

**Challenge Description**: Real-time applications are extremely latency-critical (<100ms) **Solution**:

* Model pruning and quantization reduce the amount of computation
* Edge computing deployments reduce network latency
* Hardware acceleration optimizes inference speed

**8.1.3 Environmental Adaptability**

**Challenge** Description: Recognition of different lighting and background environmental impacts **Solution**:

* Multi-environment data collection
* Robust feature design
* Environment-adaptive algorithms

**8.2 Data Challenges**

**8.2.1 Data Scarcity**

**Challenge description**: Difficulty in collecting sign language data and high cost of labeling **Solution**:

* Semi-supervised learning method
* Transfer learning techniques
* Active learning strategies

**8.2.2 Data Imbalance**

**Challenge description**: The frequency of use of vocabulary varies greatly between different sign language words **Solution**:

* Category balanced sampling
* Loss function weighting
* Data synthesis technology

**8.3 Application Challenges**

**8.3.1 User Acceptance**

**Challenge** Description: User Acceptance and Usage Habits of Technology Products **Solution**:

* User participatory design
* Progressive feature releases
* Community feedback mechanism

**8.3.2 Standardization issues**

**Challenge description**: Sign language standards are not uniform in different regions **Solution**:

* Multi-standard parallel support
* Standard conversion algorithm
* Localized adaptation

**9. Future Directions**

**9.1 Technological Development Roadmap**

**9.1.1 Short-term goals (1-2 years)**

* **Vocabulary Expansion**: Supports 200+ commonly used sign language words
* **Multi-person recognition**: Support multi-person sign language recognition at the same time
* **Mobile** deployment: Compatible with smartphones and tablets
* **Speech Synthesis**: Integrates high-quality speech output

**9.1.2 Medium-term target (3-5 years)**

* **Sentence Comprehension**: Supports full sign language sentence recognition
* **Dialogue system**: Implement two-way sign language dialogue function
* **Cross-language support**: Support for multi-country sign language systems
* **AR/VR Integration**: Deep integration into augmented reality platforms

**9.1.3 Long-term vision (5-10 years)**

* **General Sign Language Understanding**: Achieve near-human level sign language comprehension
* **Emotion** Recognition: Recognize emotional expressions in sign language
* **Personalized adaptation**: Automatically adapts to individual sign language habits
* **Ecosystem construction**: Form a complete sign language AI ecosystem

**9.2 Direction of technical optimization**

**9.2.1 Model Optimization**

* **Lightweight design**: Further compress model parameters and computational effort
* **Multimodal fusion**: Combines multiple sensor information such as vision and inertia
* **E-learning**: Enables continuous optimization of the model as it is used
* **Federated learning**: Privacy-preserving distributed model training

**9.2.2 System Optimization**

* **Cloud-edge collaboration**: The optimal combination of cloud training and edge inference
* **Automated O&M**: Automatic model updates and performance monitoring
* **Fault Tolerance**: Improve system stability and reliability
* **Security**: Protect user data and models

**9.3 Application Expansion Direction**

**9.3.1 Vertical applications**

* **Healthcare**: Rehabilitation training, health monitoring
* **Education** and training: sign language teaching, skills training
* **Entertainment and culture**: sign language games, cultural communication
* **Industrial control**: Contactless operation, safety control

**9.3.2 Platform ecological construction**

* **Developer Tools**: Provide SDKs and APIs
* **Data platform**: Build a sign language data sharing platform
* **App Store**: Build an ecosystem of sign language apps
* **Community Building**: Facilitate communication between developers and users

**10. Summary and outlook**

**10.1 Project Summary**

This project has successfully built a complete sign language recognition and translation system, and achieved the following main results:

**10.1.1 Technical Achievements**

* **Complete Framework**: An end-to-end framework from data collection to application deployment is established
* **Multi-model support**: Comparative evaluation of 7 different deep learning models was realized
* **High-precision recognition**: The best model achieves a recognition accuracy of 92.7%.
* **Real-time performance**: The average inference latency is less than 30 ms, which meets the requirements of real-time applications
* **System** Stability: Stability and reliability that have been proven through extensive testing

**10.1.2 Project Results**

* **Modular design**: Clear module division for easy maintenance and expansion
* **Standardized interface**: Unified API design supports secondary development