WaveSourceMiniResNet Architecture Summary

Overview

A custom ResNet-based CNN for predicting wave source coordinates from wave interference patterns. The network processes 128×128 wave field images and outputs (x, y) coordinates in the range [0, 127].

Stage 0:

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
Conv2d	`[B,1,128,128]`	`[B,32,64,64]`	32	7×7	2	3	Spatial: 128⟶64, Ch: 1⟶32
BatchNorm2d	`[B,32,64,64]`	`[B,32,64,64]`	32 ch	-	-	-	No change
ReLU	`[B,32,64,64]`	`[B,32,64,64]`	-	-	-	-	No change
MaxPool2d	`[B,32,64,64]`	`[B,32,32,32]`	-	3×3	2	1	Spatial: 64⟶32

Stage 1:

Block 1 (Identity Skip)

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
Conv1	`[B,32,32,32]`	`[B,32,32,32]`	32	3×3	1	1	No change
BN1+ReLU	`[B,32,32,32]`	`[B,32,32,32]`	32 ch	-	-	-	No change

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
Conv2	`[B,32,32,32]`	`[B,32,32,32]`	32	3×3	1	1	No change
BN2	`[B,32,32,32]`	`[B,32,32,32]`	32 ch	-	-	-	No change
Skip	`[B,32,32,32]`	`[B,32,32,32]`	-	-	-	-	Identity
Add+ReLU	`[B,32,32,32]`	`[B,32,32,32]`	-	-	-	-	No change

Block 2 (Identity Skip)

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
Conv1	`[B,32,32,32]`	`[B,32,32,32]`	32	3×3	1	1	No change
BN1+ReLU	`[B,32,32,32]`	`[B,32,32,32]`	32 ch	-	-	-	No change
Conv2	`[B,32,32,32]`	`[B,32,32,32]`	32	3×3	1	1	No change
BN2	`[B,32,32,32]`	`[B,32,32,32]`	32 ch	-	-	-	No change
Skip	`[B,32,32,32]`	`[B,32,32,32]`	-	-	-	-	Identity
Add+ReLU	`[B,32,32,32]`	`[B,32,32,32]`	-	-	-	-	No change

Stage 2:

Block 1 (Projection Skip)

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
Conv1	`[B,32,32,32]`	`[B,64,16,16]`	64	3×3	2	1	Spatial: 32→16, Ch: 32→64
BN1+ReLU	`[B,64,16,16]`	`[B,64,16,16]`	64 ch	-	-	-	No change
Conv2	`[B,64,16,16]`	`[B,64,16,16]`	64	3×3	1	1	No change

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
BN2	`[B,64,16,16]`	`[B,64,16,16]`	64 ch	-	-	-	No change
Skip	`[B,32,32,32]`	`[B,64,16,16]`	64	1×1	2	0	Projection
Add+ReLU	`[B,64,16,16]`	`[B,64,16,16]`	-	-	-	-	No change

Block 2 (Identity Skip)

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
Conv1	`[B,64,16,16]`	`[B,64,16,16]`	64	3×3	1	1	No change
BN1+ReLU	`[B,64,16,16]`	`[B,64,16,16]`	64 ch	-	-	-	No change
Conv2	`[B,64,16,16]`	`[B,64,16,16]`	64	3×3	1	1	No change
BN2	`[B,64,16,16]`	`[B,64,16,16]`	64 ch	-	-	-	No change
Skip	`[B,64,16,16]`	`[B,64,16,16]`	-	-	-	-	Identity
Add+ReLU	`[B,64,16,16]`	`[B,64,16,16]`	-	-	-	-	No change

Stage 3:

Block 1 (Projection Skip)

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
Conv1	`[B,64,16,16]`	`[B,128,8,8]`	128	3×3	2	1	Spatial: 16⟶8, Ch: 64⟶128
BN1+ReLU	`[B,128,8,8]`	`[B,128,8,8]`	128 ch	-	-	-	No change
Conv2	`[B,128,8,8]`	`[B,128,8,8]`	128	3×3	1	1	No change
BN2	`[B,128,8,8]`	`[B,128,8,8]`	128 ch	-	-	-	No change
Skip	`[B,64,16,16]`	`[B,128,8,8]`	128	1×1	2	0	Projection
Add+ReLU	`[B,128,8,8]`	`[B,128,8,8]`	-	-	-	-	No change

Block 2 (Identity Skip)

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
Conv1	`[B,128,8,8]`	`[B,128,8,8]`	128	3×3	1	1	No change
BN1+ReLU	`[B,128,8,8]`	`[B,128,8,8]`	128 ch	-	-	-	No change
Conv2	`[B,128,8,8]`	`[B,128,8,8]`	128	3×3	1	1	No change
BN2	`[B,128,8,8]`	`[B,128,8,8]`	128 ch	-	-	-	No change
Skip	`[B,128,8,8]`	`[B,128,8,8]`	-	-	-	-	Identity
Add+ReLU	`[B,128,8,8]`	`[B,128,8,8]`	-	-	-	-	No change

Stage 4:

Block 1 (Projection Skip)

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
Conv1	`[B,128,8,8]`	`[B,256,4,4]`	256	3×3	2	1	Spatial: 8→4, Ch: 128→256
BN1+ReLU	`[B,256,4,4]`	`[B,256,4,4]`	256 ch	-	-	-	No change
Conv2	`[B,256,4,4]`	`[B,256,4,4]`	256	3×3	1	1	No change
BN2	`[B,256,4,4]`	`[B,256,4,4]`	256 ch	-	-	-	No change
Skip	`[B,128,8,8]`	`[B,256,4,4]`	256	1×1	2	0	Projection
Add+ReLU	`[B,256,4,4]`	`[B,256,4,4]`	-	-	-	-	No change

Block 2 (Identity Skip)

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
Conv1	`[B,256,4,4]`	`[B,256,4,4]`	256	3×3	1	1	No change
BN1+ReLU	`[B,256,4,4]`	`[B,256,4,4]`	256 ch	-	-	-	No change
Conv2	`[B,256,4,4]`	`[B,256,4,4]`	256	3×3	1	1	No change
BN2	`[B,256,4,4]`	`[B,256,4,4]`	256 ch	-	-	-	No change
Skip	`[B,256,4,4]`	`[B,256,4,4]`	-	-	-	-	Identity

Layer	Input Shape	Output Shape	Filters	Kernel	Stride	Padding	Change
Add+ReLU	`[B,256,4,4]`	`[B,256,4,4]`	-	-	-	-	No change

Global Pooling & Regression Head

Layer	Input Shape	Output Shape	Neurons	Operation	Change
AdaptiveAvgPool2d	`[B,256,4,4]`	`[B,256,1,1]`	-	4×4→1×1	Spatial collapse
Flatten	`[B,256,1,1]`	`[B,256]`	-	Reshape	4D→2D
Linear	`[B,256]`	`[B,128]`	256→128	FC layer	Features: 256→128
BN1d+ReLU+Drop(0.2)	`[B,128]`	`[B,128]`	128	Regularize	20% dropout
Linear	`[B,128]`	`[B,64]`	128→64	FC layer	Features: 128→64
BN1d+ReLU+Drop(0.1)	`[B,64]`	`[B,64]`	64	Regularize	10% dropout
Linear	`[B,64]`	`[B,2]`	64→2	FC layer	Final coordinates
Sigmoid×127	`[B,2]`	`[B,2]`	-	Scale	Range [0,127]

Key Statistics

Metric	Value
Total Convolution Filters	1,952 filters
Total Parameters	~1.2M trainable parameters
Spatial Reduction	128×128 → 4×4 → 1×1 (16,384× reduction)
Channel Expansion	1 → 32 → 64 → 128 → 256 (256× expansion)
Final Output	2 coordinates in range [0, 127]

Architecture Components

Residual Blocks

- **Structure**: Two 3×3 convolutions with batch normalization
- **Skip Connections**:
  - Identity when input/output shapes match
  - 1×1 projection convolution when shapes differ
- **Pattern**: First block changes dimensions, second block refines features

Skip Connection Types

Type	When Used	Operation
Identity	Same input/output shape	Direct addition
Projection	Different input/output shape	1×1 conv + stride to match dimensions

Dimension Progression

Stage	Spatial Size	Channels	Feature Maps
Input	128×128	1	1
Stage 0	32×32	32	32
Stage 1	32×32	32	32
Stage 2	16×16	64	64
Stage 3	8×8	128	128
Stage 4	4×4	256	256
Global Pool	1×1	256	256
Output	-	2	2 coordinates

Technical Details

Convolution Parameters

- **Initial Conv**: 7×7 kernel, stride=2 for rapid spatial reduction
- **Residual Convs**: 3×3 kernels, stride=1 or 2 depending on block type
- **Projection Convs**: 1×1 kernels for efficient channel transformation

Regularization

- **Batch Normalization**: Applied after every convolution

- **Dropout**: 0.2 and 0.1 rates in regression head
- **ReLU Activation**: Applied after batch normalization

Global Pooling

- **AdaptiveAvgPool2d(1)**: Converts 4×4 feature maps to single values
- **Purpose**: Translation invariance and dimension reduction

Output Processing

- **Sigmoid Activation**: Ensures output in [0, 1] range
- **Scaling**: Multiply by 127 to get coordinates in [0, 127] range

---

Generated from WaveSourceMiniResNet architecture analysis