HDC-CNN: Cosmological Parameter Inference with Uncertainty Quantification

NeurIPS 2025 Weak Lensing Uncertainty Challenge

Hao-Chun, Liang
Student ID: 314580042 Team ID: 14

Deep Learning for Astrophysics

December 28, 2025

Mission & Challenge

Mission:

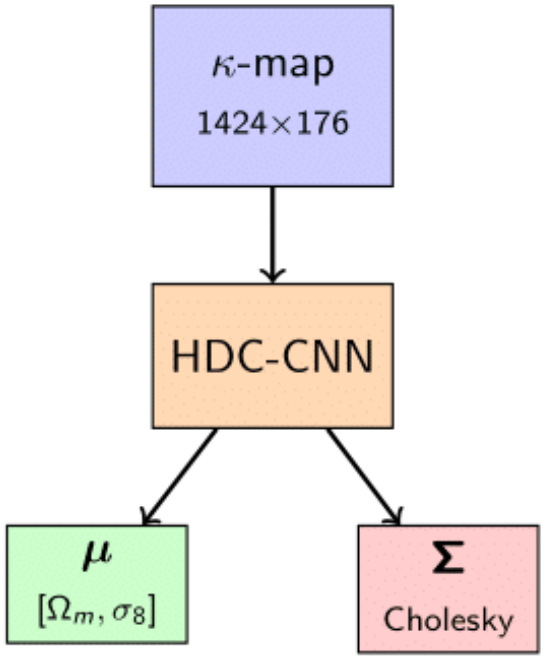
- Predict cosmological parameters (Ω_m, σ_8) from Weak Lensing Convergence Maps (κ-maps)

Challenges:

- High noise levels in observational data
- Requirement for **calibrated confidence intervals**

Solution:

- Dual-head CNN: predictions + covariance matrix
- Output: 2 means + 3 Cholesky parameters




```
# Dual-head output
mean_head = nn.Linear(64, 2)
cov_head = nn.Linear(64, 3)
```

Baseline: Startkit score = **8.70**

Data Pipeline & Validation Strategy

On-the-fly Noise Injection:

- Simulates observation conditions
- Prevents overfitting to noise

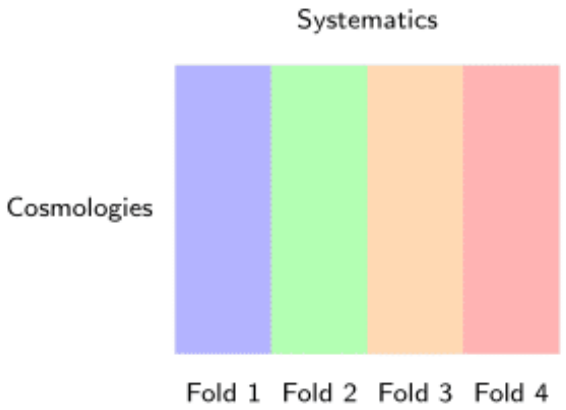
Noise: $\sigma = \sigma_{\text{shape}} / \sqrt{2 n_g \theta^2}$

σ_{shape}=0.4, n_g=30/arcmin^2, θ=2′

Configuration:

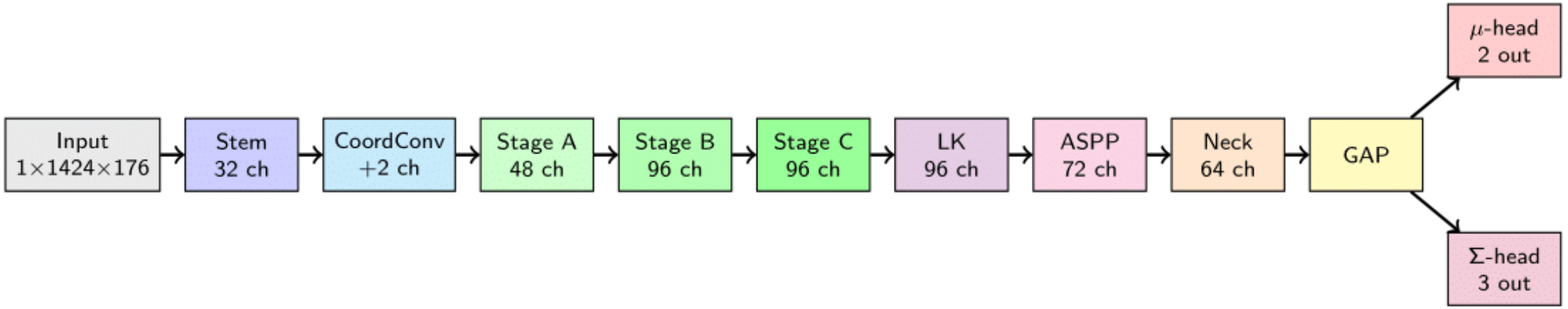
- 4 folds × 2 repeats = 8 models
- Seeds: split=42, noise=123

Systematics-aware K-Fold CV:



- Split by **systematics ID**, not random
- All 101 cosmologies in train/val
- Generalizes to unseen conditions

HDC-CNN Architecture Overview



CoordConv (Liu+ '18):

- Appends (x, y) coordinates
- Range: $[-1, 1]$
- Position-aware features

3-Stage Backbone:

- BlurPool (Zhang '19): $[1, 4, 6, 4, 1]/16$
- Horizontal stride-2 downsample
- HDC blocks per stage

Dual-Head Output:

- Mean: $[\Omega_m, \sigma_8]$
- Cov: Cholesky, softplus(L_{ii})
- $\sim$202K parameters

Key Architectural Modules

HDC Block (Wang+ '18):

- Expands receptive field without downsampling
- Dilations: $(1,2) \rightarrow (2,5) \rightarrow (2,5),(3,7)$

```
class HDCBlock(nn.Module):
  def __init__(self, ch, dilations):
    self.sep1 = SeparableConv2d(ch, d1)
    self.sep2 = SeparableConv2d(ch, d2)
    self.ls = LayerScale(ch, init=1e-5)
```

Large Kernel Branch:

- 1×63 @ dilation 4, RF: 249 pixels

ASPP Lite (Chen+ '17): (1×7), (1×15), (1×21) @ dil=1,2,2, (1×1), (63×1)

ECA (Wang+ '20):

- Adaptive kernel: $k = \lfloor (\log_2 C + 1)/2 \rfloor$
- Used in: Stage C, ASPP Lite

```
class ECA(nn.Module):
  def __init__(self, ch, gamma=2, b=1):
    k = int(abs(log2(ch)/gamma+b/gamma))
    k = k if k % 2 else k + 1
    self.conv = nn.Conv1d(1,1,k,padding=
        k//2)
```

LayerScale (Touvron+ '21) + **DropPath** (Huang+ '16):

- Per-channel scaling, init $\gamma = 10^{-5}$
- Stochastic depth for regularization

Other: WSConv2d (Qiao+ '19), GroupNorm, ThinBottleneck

Loss Function: Beta-NLL

Gaussian NLL (Baseline):

$$\mathcal{L}_{\text{NLL}} = \frac{1}{2}\left(\log \sigma^2 + \frac{(y-\mu)^2}{\sigma^2}\right)$$

Problem: Model predicts high σ^2 to minimize loss $\rightarrow$ meaningless uncertainties

Solution: Beta-NLL Loss (Seitzer+ '22)

$$w = (\sigma^2)^\beta, \quad \mathcal{L}_\beta = \frac{w}{2} \cdot \left(\frac{(y-\mu)^2}{\sigma^2} + \log \sigma^2\right)$$

Beta Controls Behavior:

- $\beta = 0$: Standard NLL; $\beta = 1$: Penalizes high-var
- **Used:** $\beta = 0.3$

```
def forward(self, mean, log_var, target)
    :
  var = torch.exp(log_var)
  sq_err = (target - mean) ** 2
  nll = sq_err/(var+eps) + log_var
  weight = var.detach() ** self.beta
  return (0.5*nll*weight).sum(1).mean()
```

Key: `var.detach()` prevents gradient through weight

Optimization & Training

AdamW (Loshchilov+ '19):

- Layer-wise LRs: mean_head 1.5×, cov_head 0.7×, backbone 1.0×
- Weight decay: 0.0005 (backbone only)

OneCycleLR (Smith '19):

- 15% warmup, cosine annealing
- max_lr: 0.0042, div_factor: 25
- Grad clip: 1.0, early stop: 17 epochs

EMA (Exponential Moving Average):

- Decay: 0.999 (≈1000 step window)
- Smoother predictions

```
class ModelEMA:
  def update(self, model):
    for name, p in model.parameters():
      self.shadow[name] = (
        0.999*self.shadow[name]+0.001*p)
```

Validation: Evaluate RAW + EMA, select best per epoch

Post-Hoc Calibration & Ensemble

Isotonic Regression (PAVA):

- Maps: predicted $\sigma^2 \rightarrow$ empirical $(y - \mu)^2$
- Applied separately to Ω_m and σ_8

```
# PAVA: merge violating blocks
while i + 1 < len(blocks):
  if block_values[i] > block_values[i
      +1]:
    merged = blocks[i] + blocks[i+1]
```

Score: $-\sum(\chi^2 + \log\sigma^2 + \lambda \cdot \text{MSE})$, λ=10^3
Calibration: 10.2365 $\rightarrow$ **10.2666**

Mixture of Gaussians (MoG):

- 8 models (4 folds $\times$ 2 repeats), $w_i = 1/8$

Ensemble Formulas

$$\mu_{\text{ens}} = \sum_i w_i \mu_i, \quad \sigma^2_{\text{ens}} = \sum_i w_i \sigma_i^2$$

```
weights = np.ones(8) / 8
mu_ens = np.sum(w * mus, axis=0)
var_cal = iso.transform(var_ens)
```

PIT Diagnostics & Conclusions

PIT (Probability Integral Transform):

- Well-calibrated $\Rightarrow$ PIT $\sim$ Uniform$(0, 1)$
- KS test validates uniformity

```
def compute_pit(y_true, mu, std):
    return norm.cdf(y_true,
                    loc=mu, scale=std)

ks_stat, p = kstest(pit, 'uniform')
```

PIT Results:

Param	Mean PIT	KS Stat
Ω_m	0.502	0.029
σ_8	0.488	0.037

Final Results:

- 4000 test predictions
- Test Score: **10.38**
- OOF CV: $\sim$10.27

Key Takeaways:

1. **HDC architecture** captures multi-scale cosmological features
2. **Beta-NLL** (β=0.3) stabilizes uncertainty learning
3. **Isotonic calibration** corrects miscalibration
4. **MoG ensemble** reduces variance

Appendix: Technical Details

Reproducibility:

```
torch.manual_seed(42)
torch.cuda.manual_seed_all(42)
np.random.seed(42)
cudnn.deterministic = True
cudnn.benchmark = False
```

Memory Optimization:

- Gradient checkpointing enabled
- AMP with float16
- Batch size: 36 (∼2GB VRAM)

Key Files:

Component	File
Architecture	`models/hdc_cnn.py`
Layers	`models/layers.py`
Loss	`training/losses.py`
Trainer	`training/trainer.py`
K-Fold	`data/splits.py`
Calibration	`utils/isotonic.py`

Model Registry Pattern:

```
@ModelRegistry.register("hdc_cnn")
class HDCCNN(BaseModel): ...
```

Appendix: Technical Details

RANK	USERNAME	SUBMISSION ID	SCORE	MSE	R^2	COVERAGE	METHOD NAME
1	cmbagent (team)	424324	11.7321	0.1022	0.8958	0.7056	tomborrett_cmbagent
2	eiffl (team)	424310	11.6612	0.1028	0.8952	0.7095	b-remy_e36uxobt
3	shubhojit	417744	11.6192	0.1025	0.8956	0.6583	CNNv48_
4	THUML (team)	424064	11.5209	0.1064	0.8916	0.6907	F1
5	adscft	424300	11.5142	0.1056	0.8924	0.7279	cnnv11152
6	piyush555	394590	11.4681	0.1077	0.8902	0.7103	dns_l
7	jhu_suicee	423555	11.4590	0.1077	0.8903	0.6512	STILI
8	azhang81	424251	11.4192	0.1098	0.8882	0.7017	m6
9	mmayr	424022	11.2759	0.1133	0.8846	0.7080	NL128_LD512_NB2_NSA6_NH16_LR2e4_BS32_NP200_VPR0
10	jagoncalves	418273	11.2437	0.1160	0.8818	0.6835	20251008_173243
11	DOT (team)	422646	11.2203	0.1160	0.8818	0.7016	CNN
12	andry834	422647	11.1691	0.1161	0.8817	0.7097	25-11-14-03-31
13	hnayak	413583	11.1505	0.1177	0.8801	0.7130	updated3
14	Yuedong (team)	422666	11.1385	0.1197	0.8780	0.6775	25-11-13-21-12
15	deleted_user_34276	423158	11.1220	0.1174	0.8804	0.6903	777777777
16	uccaeid	424311	11.1220	0.1174	0.8804	0.6903	000000000
17	farhankhan	423146	11.0931	0.1188	0.8790	0.6871	777777
18	ahajighasem	422359	10.9727	0.1219	0.8758	0.7026	444
19	MaDESYn (team)	402355	10.9509	0.1235	0.8741	0.7201	argustarv1
20	young_hee	409010	10.9447	0.1225	0.8752	0.7037	model8
21	mshamani	422137	10.9268	0.1232	0.8745	0.6876	4444
22	sakib	375549	10.8968	0.1215	0.8762	0.7216	CNN
23	satyamk	424302	10.8300	0.1283	0.8693	0.6970	tta
24	hao_chun_liang	403355	10.7856	0.1265	0.8711	0.6934	ensemble_calibrated_exp6_20251027_23060

References

- Liu et al., "An Intriguing Failing of CNNs and the CoordConv Solution," NeurIPS 2018
- Zhang, "Making Convolutional Networks Shift-Invariant Again," ICML 2019
- Wang et al., "Understanding Convolution for Semantic Segmentation," WACV 2018
- Chen et al., "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," TPAMI 2017
- Wang et al., "ECA-Net: Efficient Channel Attention for Deep CNNs," CVPR 2020
- Touvron et al., "Going Deeper with Image Transformers," ICCV 2021
- Huang et al., "Deep Networks with Stochastic Depth," ECCV 2016
- Qiao et al., "Micro-Batch Training with Batch-Channel Normalization and Weight Standardization," arXiv 2019
- Seitzer et al., "On the Pitfalls of Heteroscedastic Uncertainty Estimation with Probabilistic Neural Networks," ICLR 2022
- Loshchilov & Hutter, "Decoupled Weight Decay Regularization," ICLR 2019
- Smith & Topin, "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates," 2019

Thank You

Questions?

Test Score: **10.38** (Baseline: 8.70)

Challenge: https://reurl.cc/zKoam7

Startkit: https://reurl.cc/6b9mpV

Dataset: https://reurl.cc/qKoWzR