Neural Networks for Predicting Algorithm Runtime Distributions

Katharina Eggensperger, Marius Lindauer & Frank Hutter

Paper ID #2772

Motivation

Algorithm portfolios yield state-of-the-art performance for SAT, ASP, Planning, …

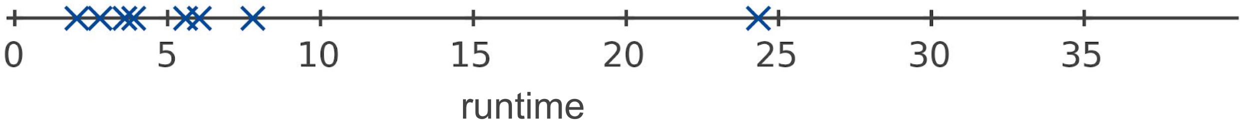
→ to build these we can make use of runtime predictions

Other applications:

- Optimal restarts
- Algorithm selection
- Algorithm configurations

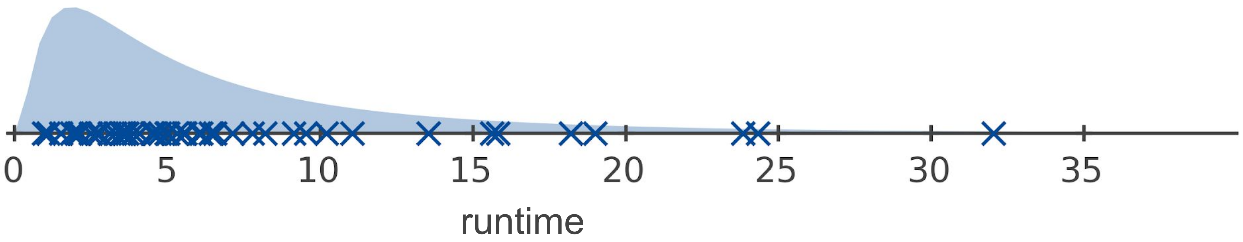
Describing the Runtime of an Algorithm?

```
solve(instance, seed):
    # do something
    return solution, runtime
```



Describing the Runtime of an Algorithm?

```
solve(instance, seed):
     # do something
     return solution, runtime
```



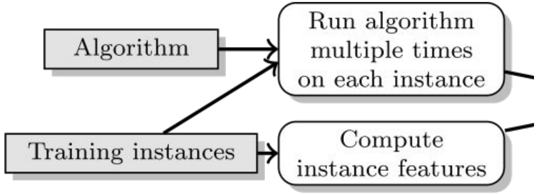
Contributions

1 Study how to **predict parametric RTDs**

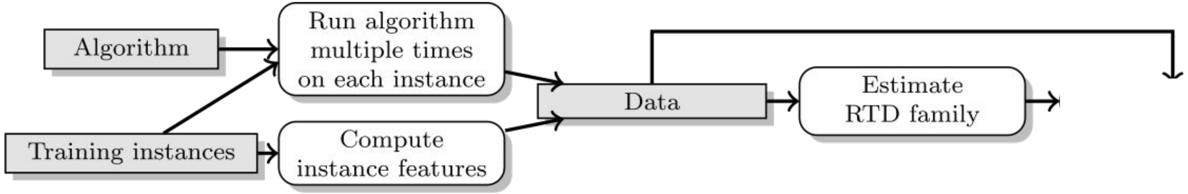
2 Propose **DistNet**, a practical neural network for predicting RTDs

3 Evaluate DistNet and show that it can **learn from only a few samples per instance**

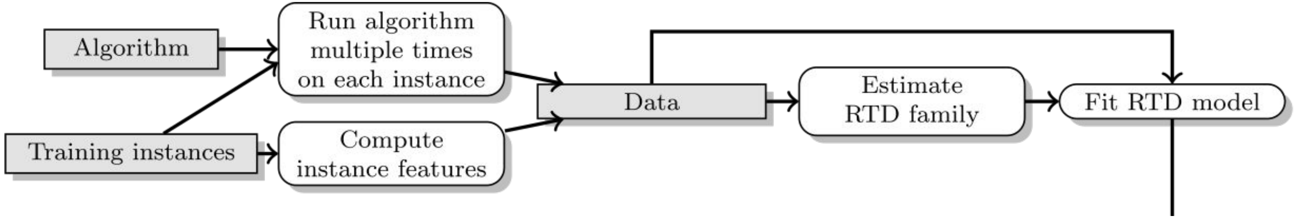
Typical Pipeline for Runtime prediction



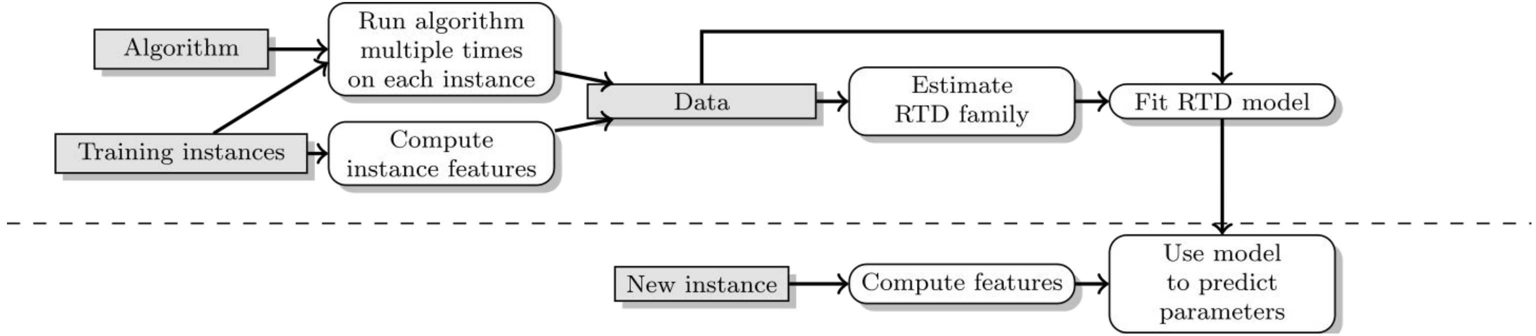
Typical Pipeline for Runtime prediction



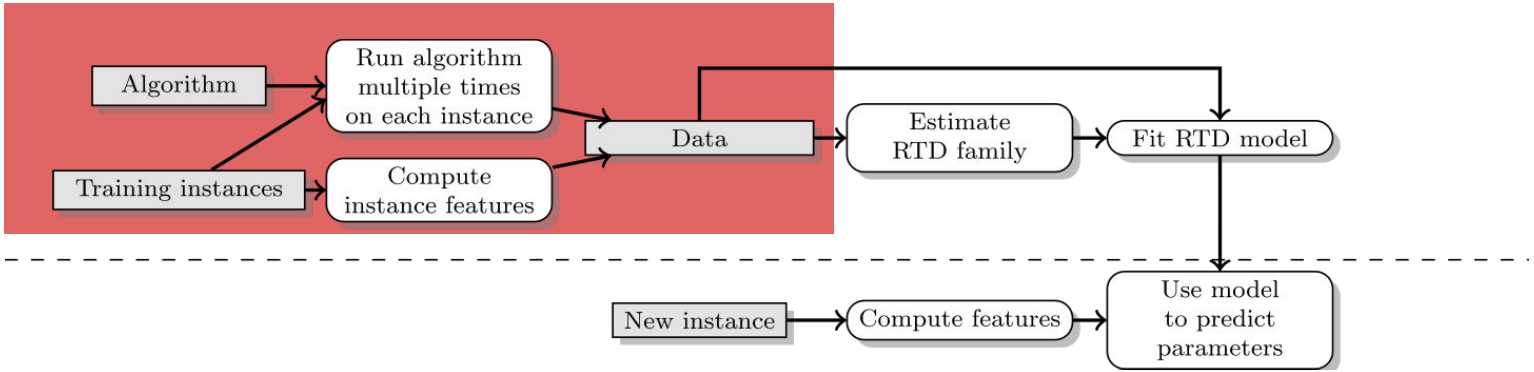
Typical Pipeline for Runtime prediction



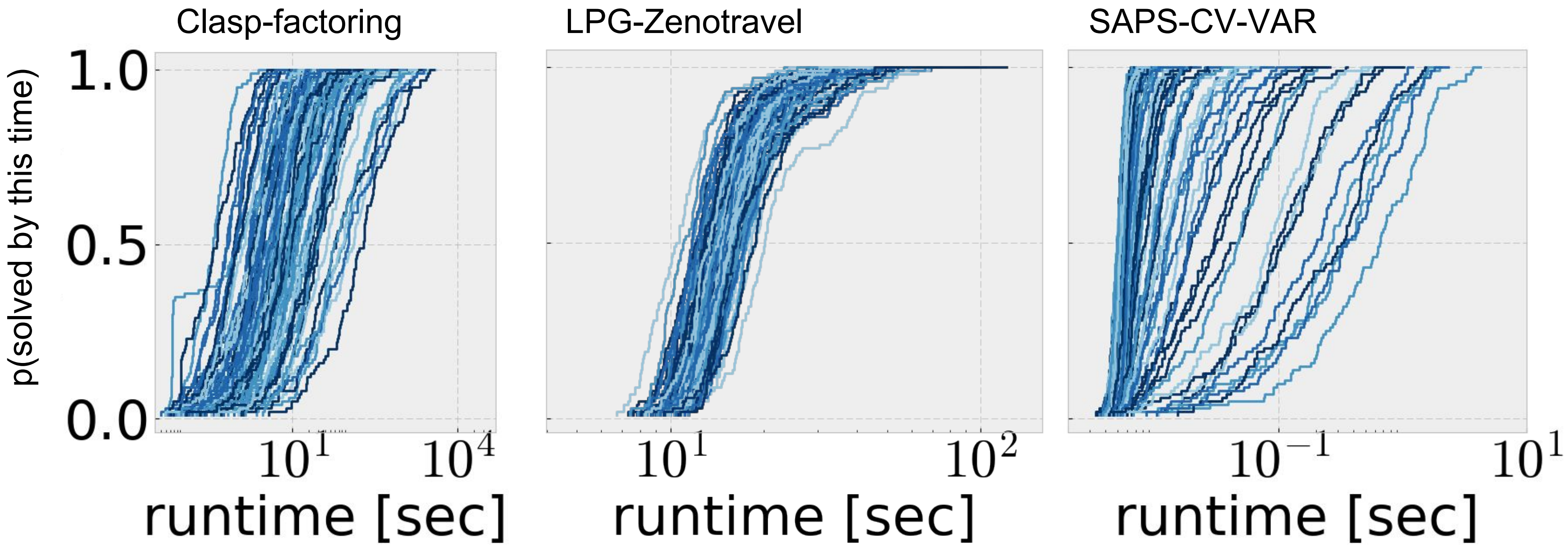
Typical Pipeline for Runtime prediction



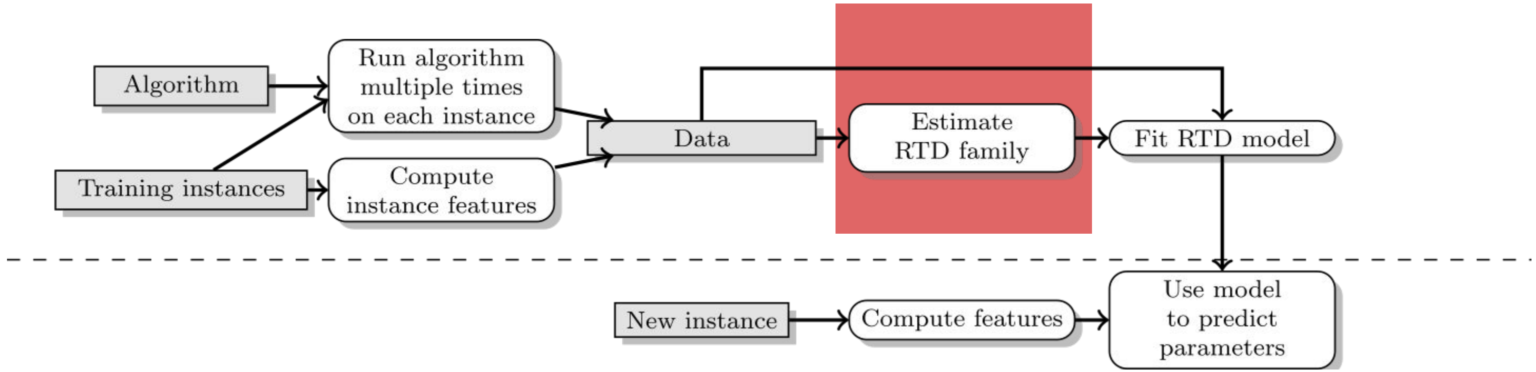
Typical Pipeline for Runtime prediction



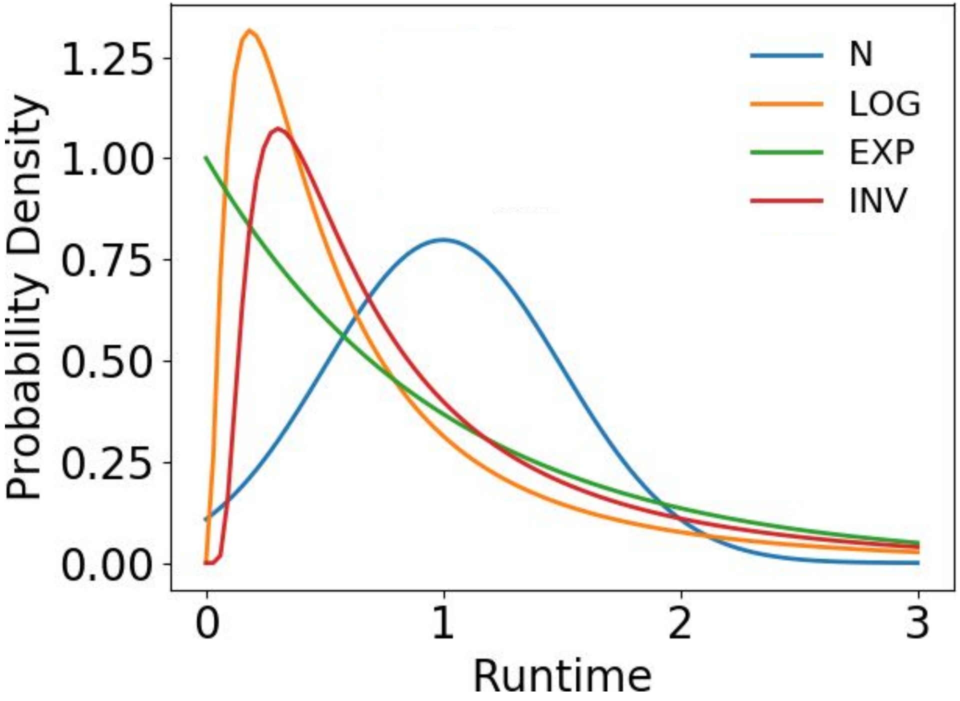
Empirical RTDs



Typical Pipeline for Runtime prediction



Considered Parametric Distribution



Distribution	Param.
Normal (N)	μ, σ
Lognormal (LOG)	s, σ
Exponential (EXP)	β
Inverse Normal (INV)	μ, λ

Quantifying the Quality of Runtime Distributions

$$\mathcal{L}_{\mathcal{D}}(\underbrace{\boldsymbol{\theta}}_{\text{distribution parameter}} \mid \underbrace{t(\pi)_1, \ldots, t(\pi)_k}_{\text{observed runtimes}}) = \prod_{i=1}^{k} p_{\mathcal{D}}(t(\pi)_i \mid \boldsymbol{\theta}) \qquad (1)$$

Quantifying the Quality of Runtime Distributions

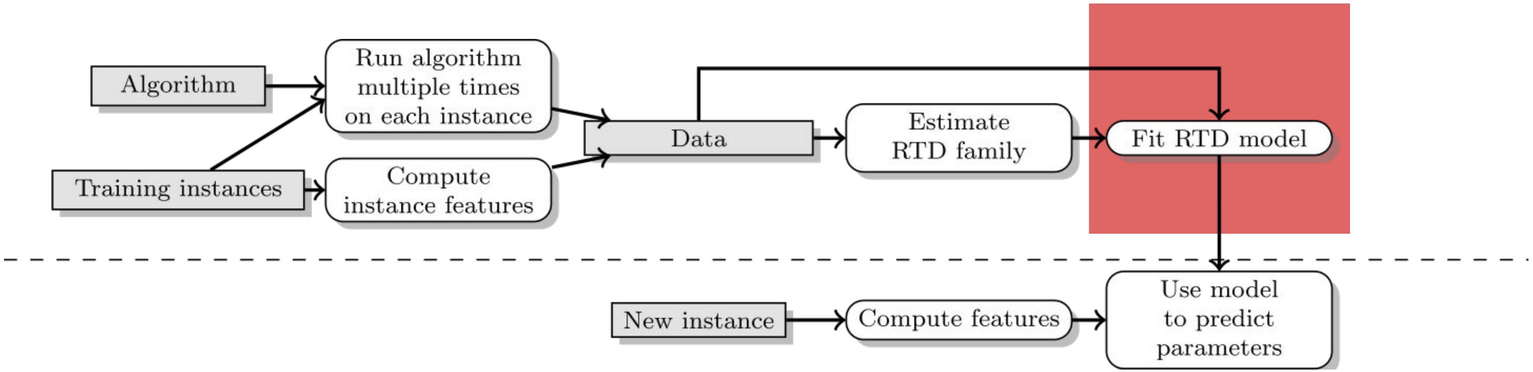
$$\mathcal{L}_{\mathcal{D}}(\underline{\boldsymbol{\theta}} \mid \underline{t(\pi)_1, \ldots, t(\pi)_k}) = \prod_{i=1}^{k} p_{\mathcal{D}}(t(\pi)_i \mid \boldsymbol{\theta}) \tag{1}$$

observed runtimes

distribution parameter

$$-\log \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta} \mid t(\pi)_1, \ldots, t(\pi)_k) = -\sum_{i=1}^{k} \log p_{\mathcal{D}}(t(\pi)_i \mid \boldsymbol{\theta}) \tag{2}$$

Typical Pipeline for Runtime prediction



Predicting multiple Runtime Distributions

Option 1

For each training instance
→ fit the parametric distribution's parameter on observed runtimes.

Then for all training instances, for each distribution parameter:
fit a model

Predicting multiple Runtime Distributions

Option 1

For each training instance
→ fit the parametric distribution's parameter on observed runtimes.

Then for all training instances, for each distribution parameter:
fit a model

Problematic, because models
- can only be as good as each fitted distribution
- do not know about interaction between their outputs
- typically minimize loss in the parameter space

Predicting multiple Runtime Distributions

Option 2

For each training instance
→ fit the parametric distribution's parameter on observed runtimes.

Then for all training instances, ~~for each distribution parameter~~:

fit a model with multiple outputs

Problematic, because model

- can only be as good as each fitted distribution
- does not know about interaction between their outputs
- typically minimizes loss in the parameter space

Predicting multiple Runtime Distributions

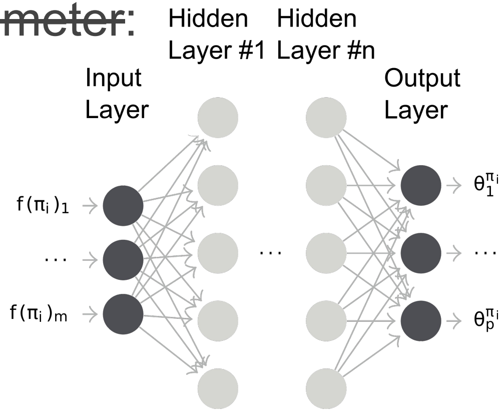
DistNet

~~For each training instance~~

~~→ fit the parametric distribution's parameter on observed runtimes.~~

Then for all training instances, ~~for each distribution parameter~~:

fit a neural network using negative log-likelihood as a loss function



Results

Scenario	dist	iRF	mRF	DistNet
Saps-CV-VAR	LOG	0.99	-0.29	**-0.52**

We compared

- DistNet
- independent Random Forests (iRF)
- multi-output Random Forests (mRF)

on 7 scenarios from SAT solving and AI planning.

Figure: Averaged negative log-likelihood. Smaller values are better.

Results

Scenario	dist	iRF	mRF	DistNet
Saps-CV-VAR	LOG	0.99	-0.29	**-0.52**
	INV	0.22	-0.09	**-0.54**

Figure: Averaged negative log-likelihood. Smaller values are better.

We compared

- DistNet
- independent Random Forests (iRF)
- multi-output Random Forests (mRF)

on 7 scenarios from SAT solving and AI planning.

Results

Scenario	dist	iRF	mRF	DistNet
Saps-CV-VAR	LOG	0.99	-0.29	**-0.52**
	INV	0.22	-0.09	**-0.54**
Clasp-factoring	INV	-0.04	-0.09	**-0.16**
	LOG	-0.14	-0.13	**-0.14**
[...]				
LPG-Zenotravel	LOG	**-0.85**	-0.84	**-0.85**
	INV	-0.72	-0.80	**-0.84**

Figure: Averaged negative log-likelihood. Smaller values are better.

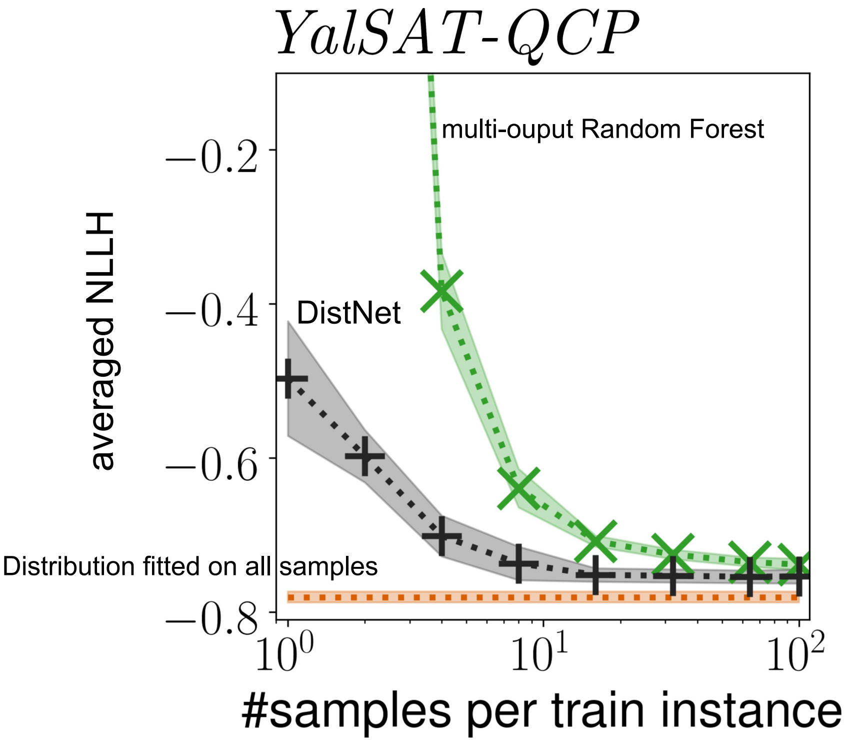
We compared

- DistNet
- independent Random Forests (iRF)
- multi-output Random Forests (mRF)

on 7 scenarios from SAT solving and AI planning.

→ Predicting parameters for RTDs is possible
→ Joint predictions work better
→ DistNet provides more robust predictions which are often better than those of competitors

DistNet on a Low Number of Observations



Wrap-Up

We have proposed DistNet, which

- **jointly learns** distribution parameters
- directly optimizes the **loss function of interest**
- performs well even if **only few observations per instance** are available

Wrap-Up

We have proposed DistNet, which

+ **jointly learns** distribution parameters
+ directly optimizes the **loss function of interest**
+ performs well even if **only few observations per instance** are available

Open Questions:

- How to automatically determine a well fitting distribution family?
- How to handle heterogeneous datasets?

Wrap-Up

We have proposed DistNet, which

+ **jointly learns** distribution parameters
+ directly optimizes the **loss function of interest**
+ performs well even if **only few observations per instance** are available

Open Questions:

- How to automatically determine a well fitting distribution family?
- How to handle heterogeneous datasets?

Code and data: https://www.automl.org/distnet/