DeepHyper

ALCF Computational Performance Workshop, May 2020

Argonne NATIONAL LABORATORY

Romain Egele

Engineering student in Computer Science & Applied Mathematics, specialized in Software and Artificial Intelligence at ENSEEIHT (Toulouse, France) & École Polytechnique (Paris)

Prasanna Balaprakash

Computer Scientist, Argonne National Laboratory [Project Lead]

Team


Misha Salim
ALCF


Stefan Wild
MCS


Venkat Vishwanath
ALCF


Taylor Childers
ALCF


Tom Uram
ALCF


Elise Jennings
ALCF


Romit Maulick
ALCF

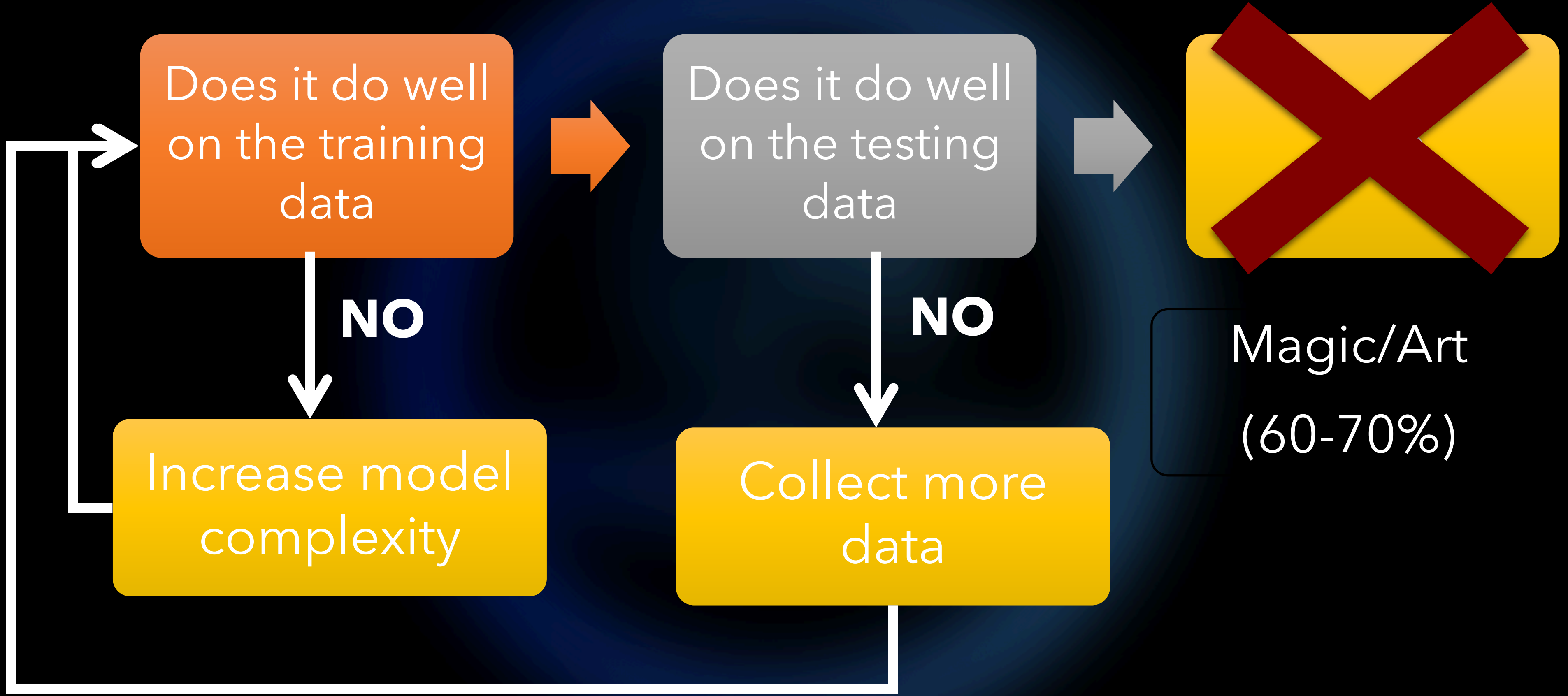

Bethany Lusch
ALCF

Contributors

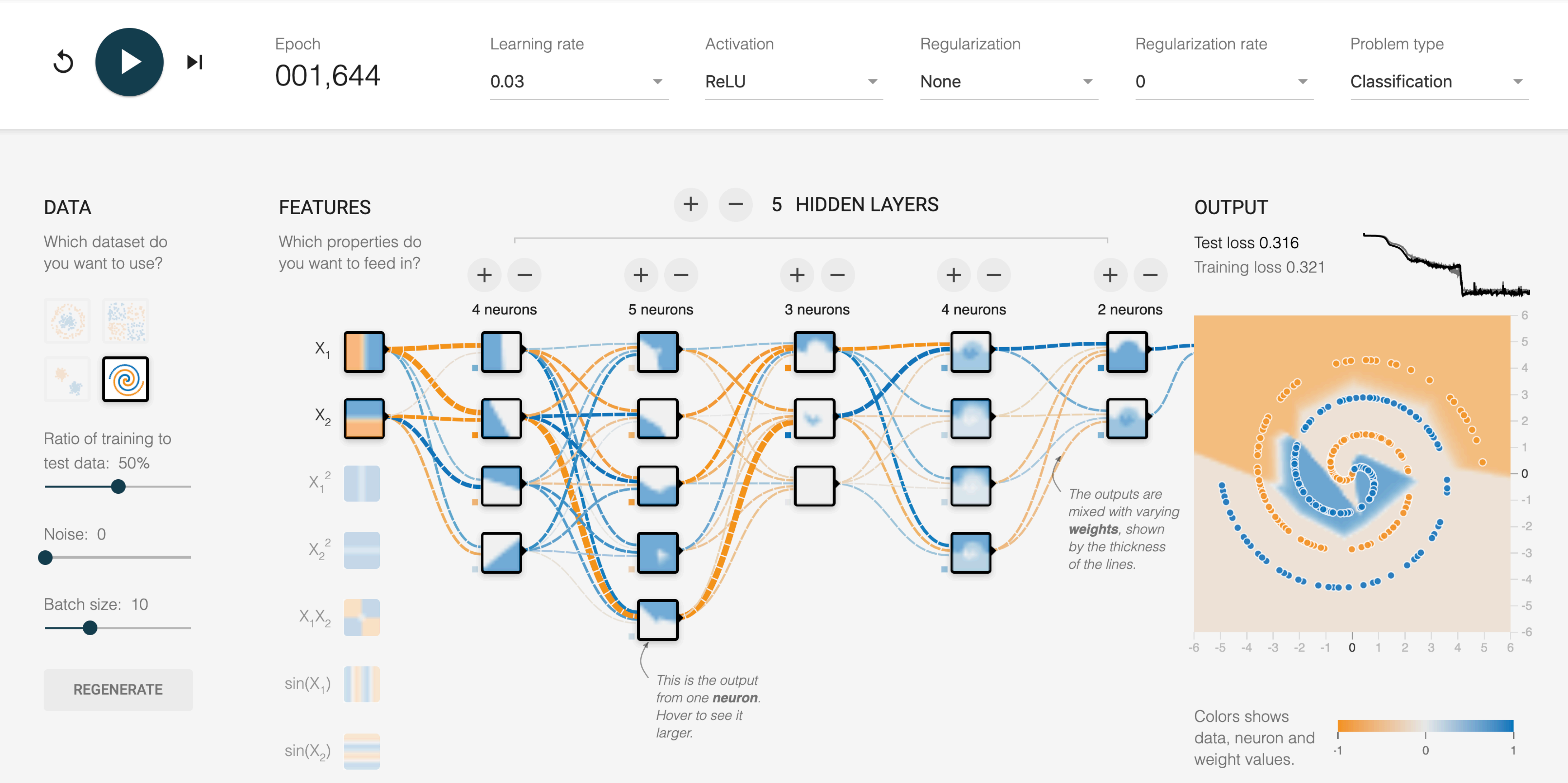

Kyle Gerard Felker
ALCF


Matthieu Dorier
MCS

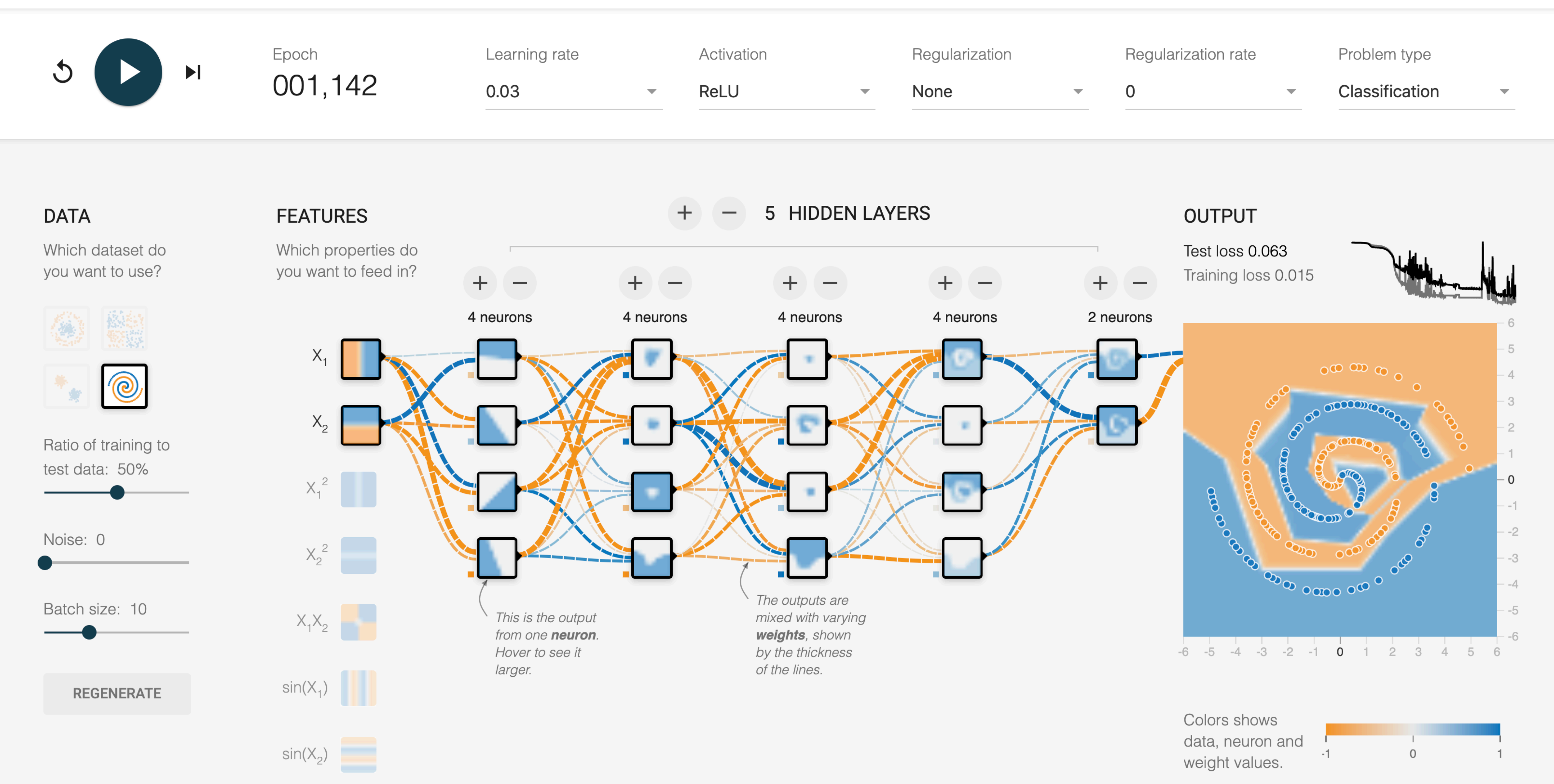
The way of Deep neural networks



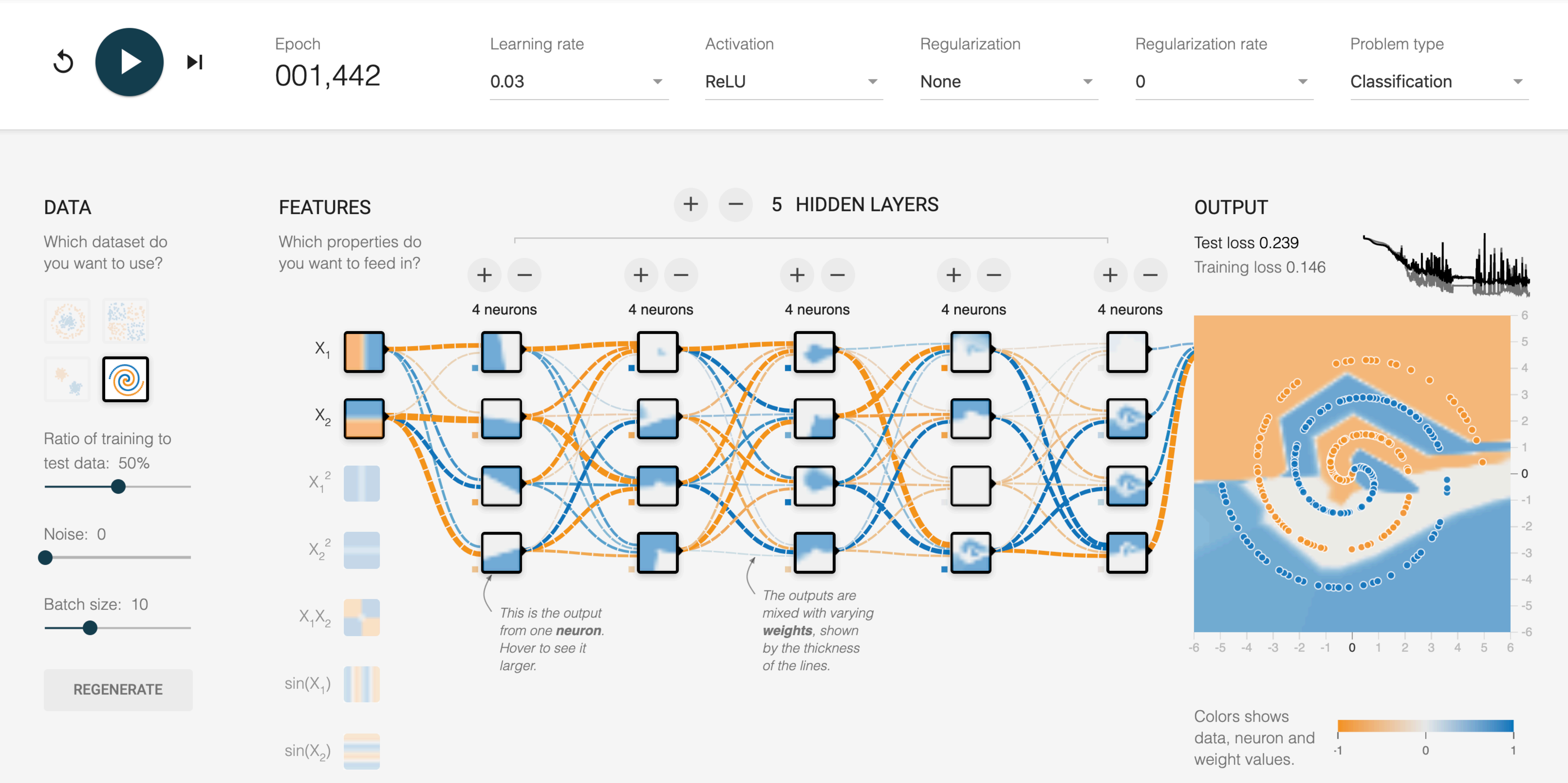
Ng A., Challenges of Deep Learning, GTC, 2015



https://playground.tensorflow.org/



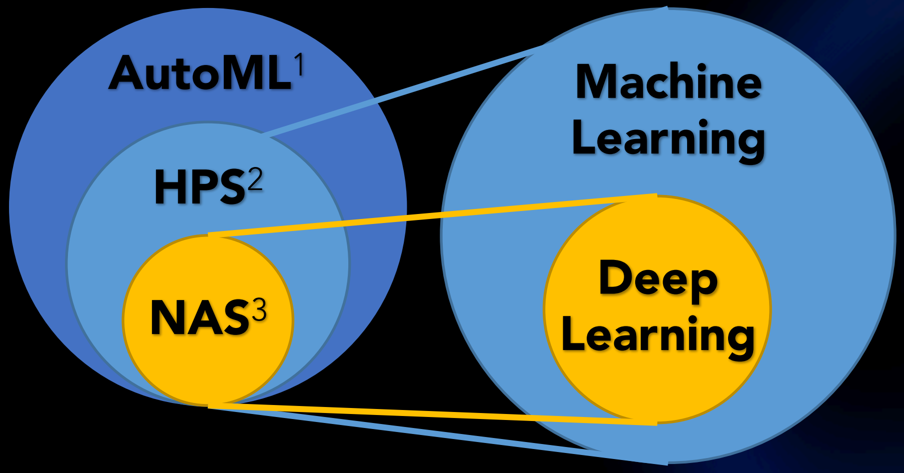
https://playground.tensorflow.org/



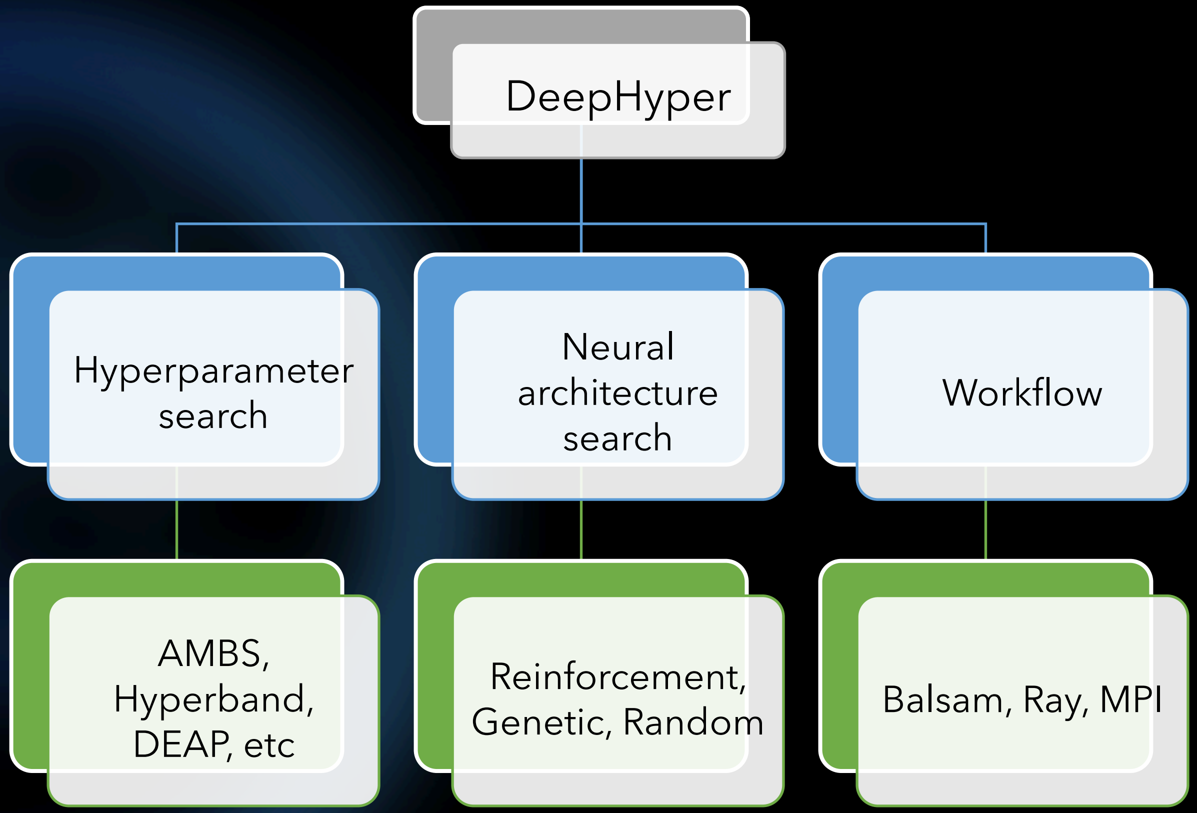
https://playground.tensorflow.org/

A scalable automated machine learning (AutoML) package for developing deep neural networks

https://github.com/deephyper/deephyper

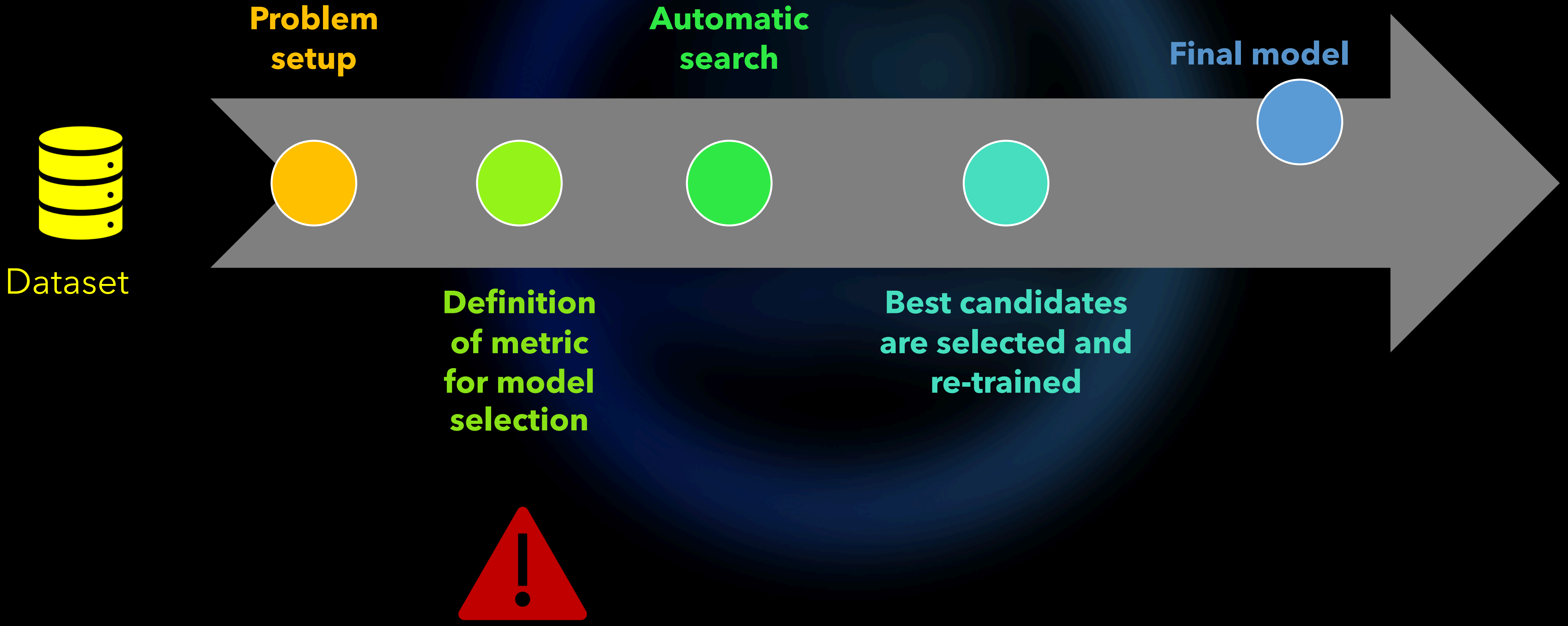


[1]Automated Machine Learning
[2]Hyperparameter Search
[3]Neural Architecture Search



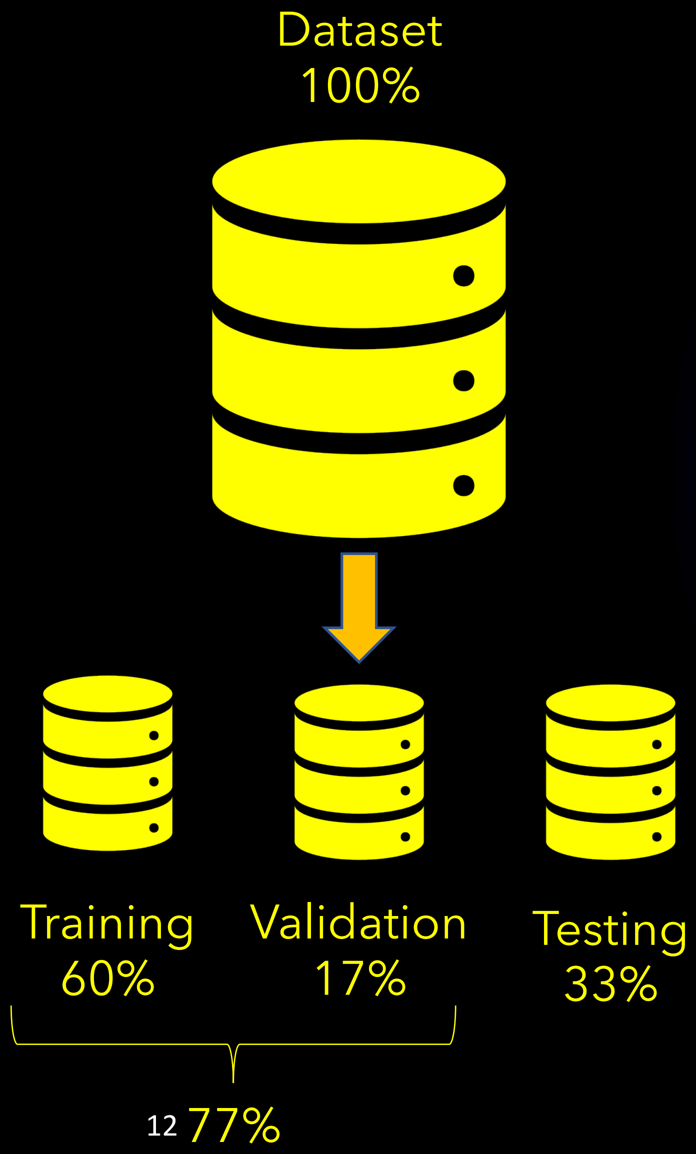
The DeepHyper workflow

The DeepHyper workflow



Argonne NATIONAL LABORATORY

Problem setup



load_data.py

```
def load_data():
    …
    Return Training, Validation
```

problem.py

```
Problem = HpProblem(seed=42)

Problem.add_hyperparameter(alpha, (0.0, 1.0))

Problem.add_starting_point(alpha=0.01)
```

Argonne NATIONAL LABORATORY

Definition of metric for model selection

run.py

```
def run(configuration):
    set_random_state(seed)
    training, validation = load_data()
    model = create_model(configuration)
    model.fit(training)
    score = model.evaluate(validation, metric)
    objective = compute_objective(score)
    return objective
```

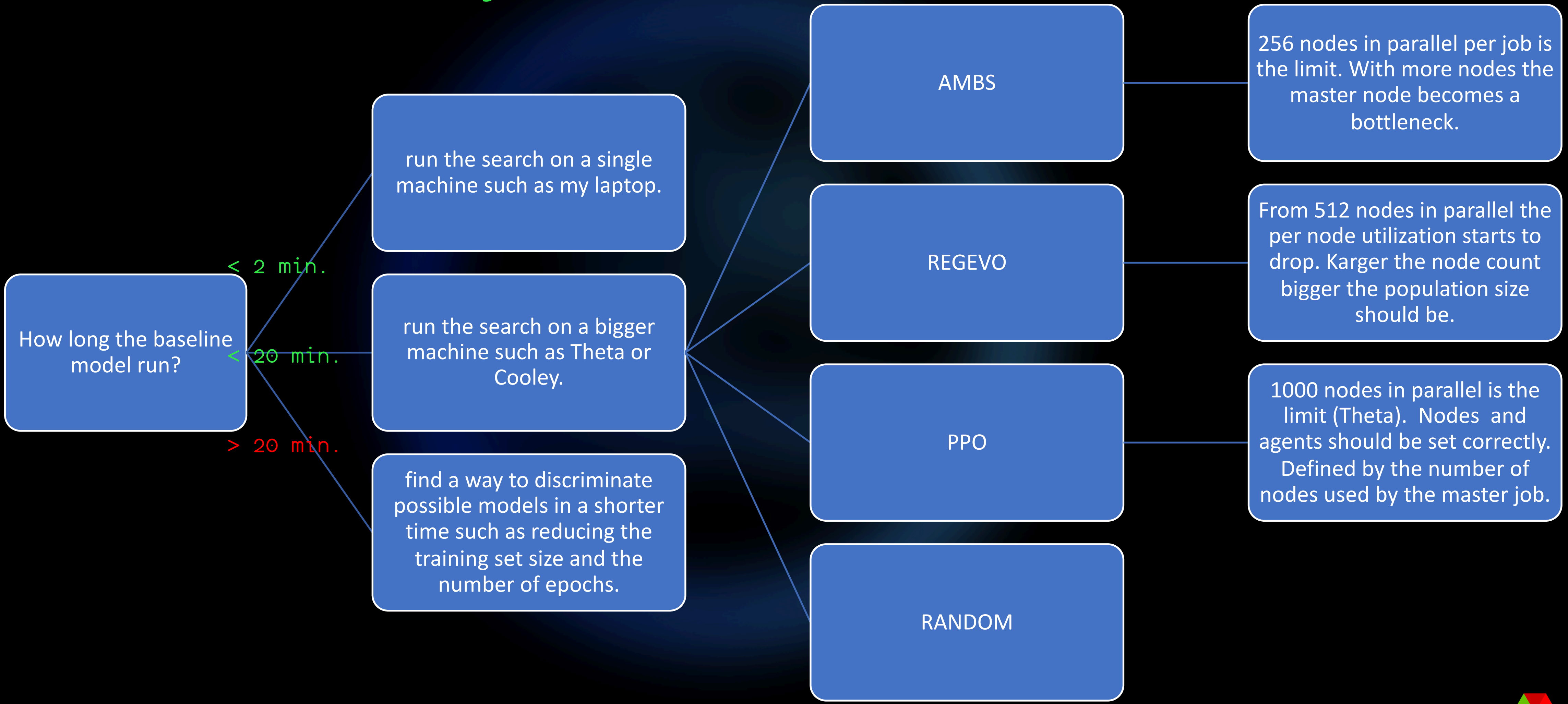
Automatic search

How to choose my search?

HPS	if you have a reasonable baseline model (NN or scikit-learn based models)
AutoML for scikit-learn	if you do not reasonable baseline scikit-learn model (such as random forest, xgboost etc)
NAS	If you do not have a reasonable NN model (make sure it is NAS compatible)

Automatic Search

How to scale or run my search?



Best candidates selection and retraining

With the « deephyper-analytics » commande line you can analyze the results of your search. See if the outcome is meaningful then rank the evaluated models and select the top-k. In the case of neural networks you can launch a post-training procedure to train the top-k models to their limits with a greater number of epochs for instance.

Final model

You should not have used the « testing dataset » yet.
Evaluate the best model you have on it and you will have its final performance.

Good Practices (alias Zen of DeepHyper)

1. Always have a baseline model before starting using DeepHyper (see "`deephyper.baseline`")

2. Always try it on your local machine before running experiments at scale

3. Always try it in debug queue before running experiments at scale

I - General Hyperparameter Search (HPS)

II - Hyperparameter Search for AutoML

III - Neural Architecture Search (NAS)

I

General Hyperparameter Search (HPS)

A bilevel optimization framework

Lower-level problem: Training data

$$\text{solve} \quad \underset{w}{\text{minimize}}\ \text{err}_T\left(\left[\mathcal{X}_{\mathcal{A}}, \mathcal{X}_{\mathcal{P}}\right]; \mathcal{T}; w\right)$$

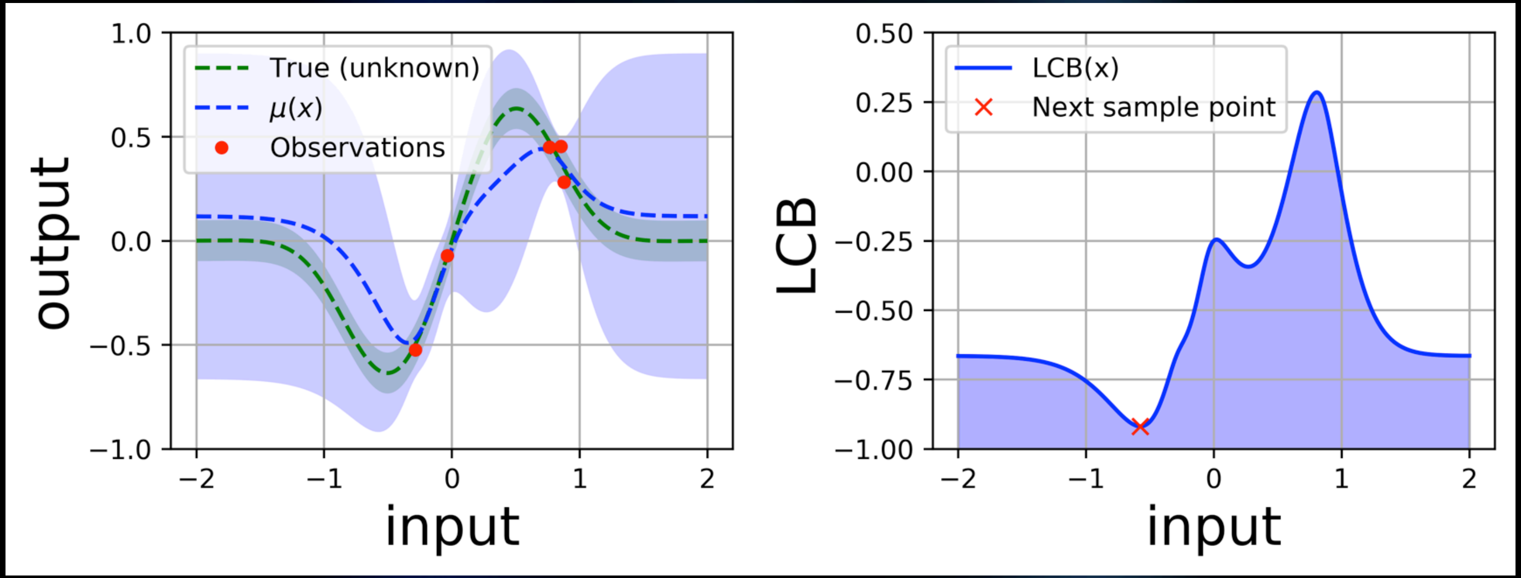
Upper-level problem: Validation data

$$\text{solve} \quad \underset{\mathcal{X}_{\mathcal{A}}, \mathcal{X}_{\mathcal{P}}}{\text{minimize}}\ \text{err}_V\left(\left[\mathcal{X}_{\mathcal{A}}, \mathcal{X}_{\mathcal{P}}\right]; \mathcal{V}; w^*\left[\mathcal{X}_{\mathcal{A}}, \mathcal{X}_{\mathcal{P}}\right]\right)$$

Architecture space Hyperparameter space

Bayesian optimization

$$LCB(x, \beta) = \mu(x) - \beta \times \sigma(x)$$



Argonne NATIONAL LABORATORY

Problem example

```python
from deephyper.problem import HpProblem


Problem = HpProblem()
Problem.add_dim("epochs", (5, 500))
Problem.add_dim("nunits_l1", (1, 1000))
Problem.add_dim("nunits_l2", (1, 1000))
Problem.add_dim("activation_l1", ["relu", "elu", "selu", "tanh"])
Problem.add_dim("activation_l2", ["relu", "elu", "selu", "tanh"])
Problem.add_dim("batch_size", (8, 1024))
Problem.add_dim("dropout_l1", (0.0, 1.0))
Problem.add_dim("dropout_l2", (0.0, 1.0))
```

II

Hyperparameter Search for AutoML

Problem example

```
import numpy as np
from deephyper.search.hps.automl.classifier import autosklearn1


def load_data():
    from sklearn.datasets import load_breast_cancer


    X, y = load_breast_cancer(return_X_y=True)
    print(np.shape(X))
    print(np.shape(y))
    return X, y



def run(config):
    return autosklearn1.run(config, load_data)
```


Argonne NATIONAL LABORATORY

Custom auto-sklearn?

```python
import ConfigSpace as cs
from deephyper.problem import HpProblem


Problem = HpProblem(seed=45)

classifier = Problem.add_hyperparameter(
    name="classifier",
    value=["RandomForest", "Logistic", "AdaBoost", "KNeighbors", "MLP", "SVC", "XGBoost"],
)

# n_estimators
n_estimators = Problem.add_hyperparameter(
    name="n_estimators", value=(1, 2000, "log-uniform")
)

cond_n_estimators = cs.OrConjunction(
    cs.EqualsCondition(n_estimators, classifier, "RandomForest"),
    cs.EqualsCondition(n_estimators, classifier, "AdaBoost"),
)


Problem.add_condition(cond_n_estimators)
…
```

[Documentation of ConfigSpace: automl.github.io/ConfigSpace](https://automl.github.io/ConfigSpace)

Model selection advice

```
from sklearn.model_selection import Kfold
from deephyper.search.nas.model.preprocessing import minmaxstdscaler

kf = KFold(n_splits=10, random_state=42, shuffle=True)
cross_score = []


for train_index, valid_index in kf.split(X):
     X_train, X_valid = X[train_index], X[valid_index]
     y_train, y_valid = y[train_index], y[valid_index]


     sm = SMOTE(random_state=seed, k_neighbors=smote_k_neighbors, n_jobs=4)
     X_train, y_train = sm.fit_resample(X_train, y_train)

     scaler = minmaxstdscaler()
     X_train = scaler.fit_transform(X_train)
     X_valid = scaler.transform(X_valid)
     …
      score = model.evaluate(X_valid, metric)
     cross_scores.append(score)
     …
     return mean(cross_scores)
```

Avoid lucky findings

For unbalanced classes

Distributed computation on CPU

Visualize your results #1

Visual Studio Code + Rainbow CSV

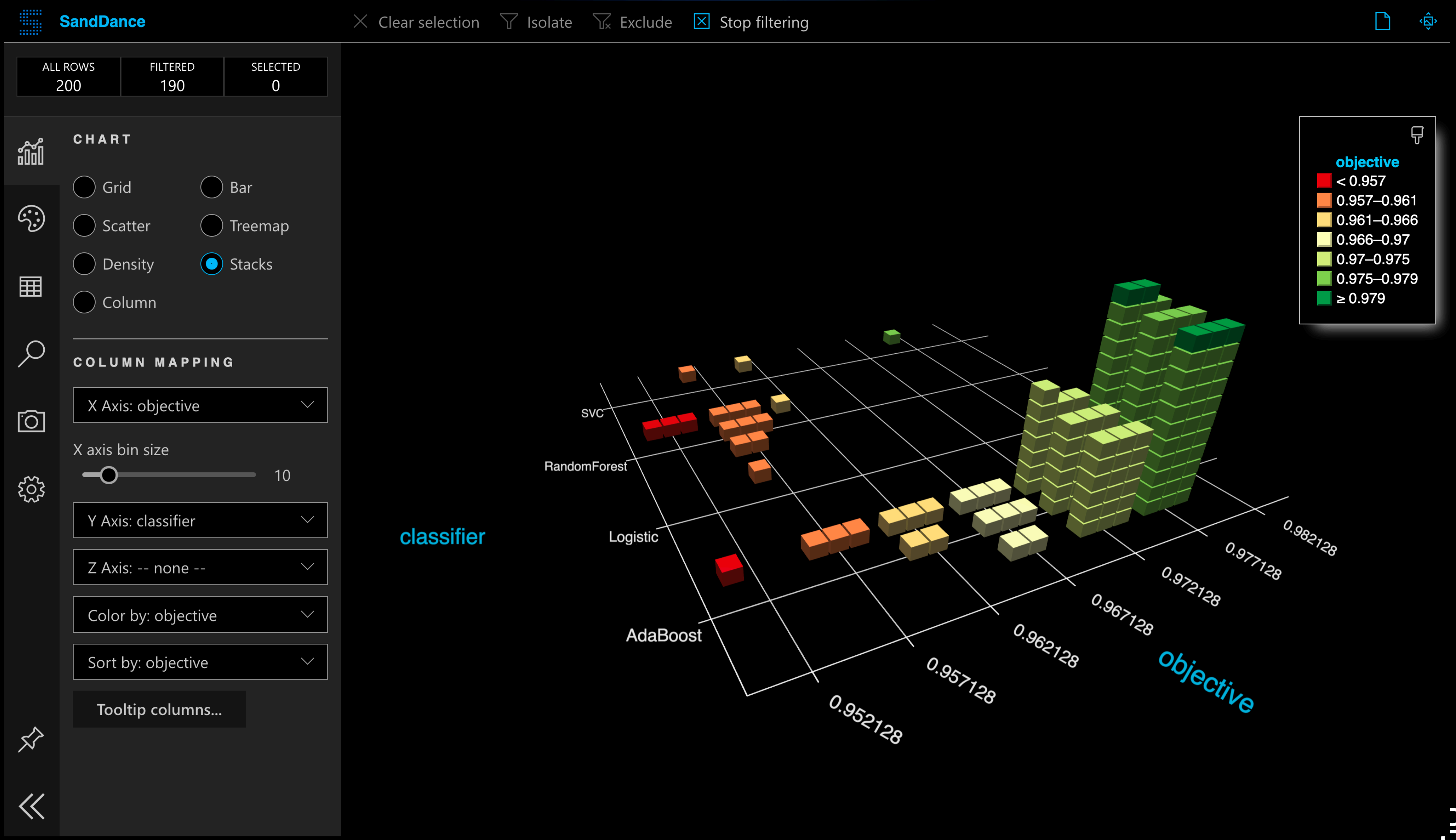
```
classifier,C,alpha,kernel,max_depth,n_estimators,n_neighbors,gamma,objective,elapsed_sec
AdaBoost,nan,nan,NA,nan,187,nan,nan,0.9627659574468085,3.8781380653381348
AdaBoost,nan,nan,NA,nan,19,nan,nan,0.9627659574468085,7.370249271392822
SVC,0.910144037187624,nan,linear,nan,nan,nan,nan,0.9574468085106383,11.247097969055176
Logistic,0.056704414597599125,nan,NA,nan,nan,nan,nan,0.9574468085106383,15.790768146514893
AdaBoost,nan,nan,NA,nan,1662,nan,nan,0.973404255319149,22.461848974227905
RandomForest,nan,nan,NA,64,561,nan,nan,0.9574468085106383,26.977345943450928
RandomForest,nan,nan,NA,15,1812,nan,nan,0.9574468085106383,33.18859791755676
```

Visualize your results #2

Visual Studio Code + SandDance

Visualize your results #3

Visual Studio Code + SandDance

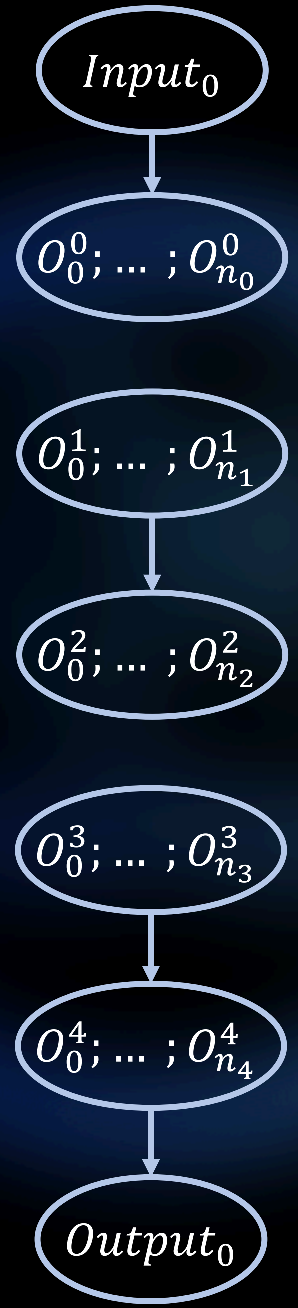


III

Neural Architecture Search (NAS)

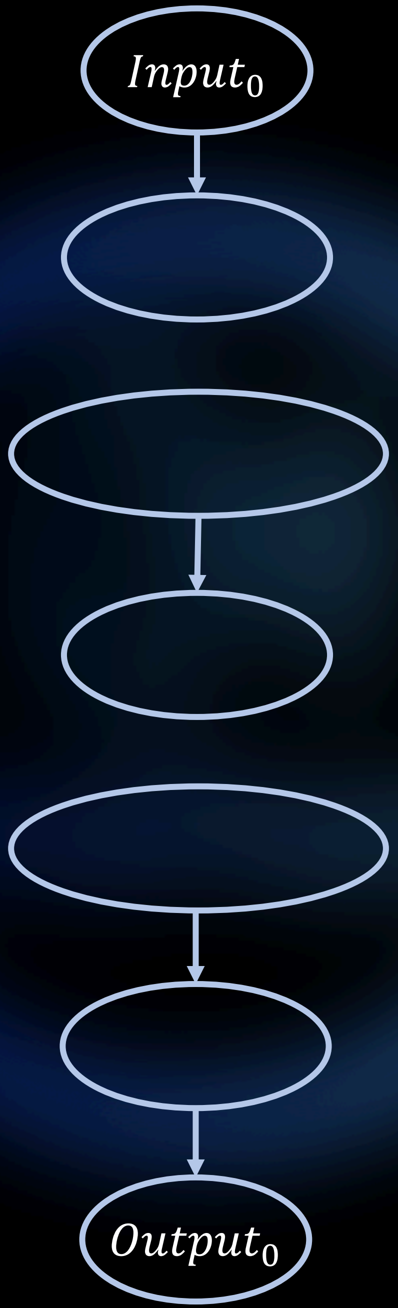
The NAS Search Space

A discrete space embedded as a directed graph where each nodes represents a choice between differents operations.



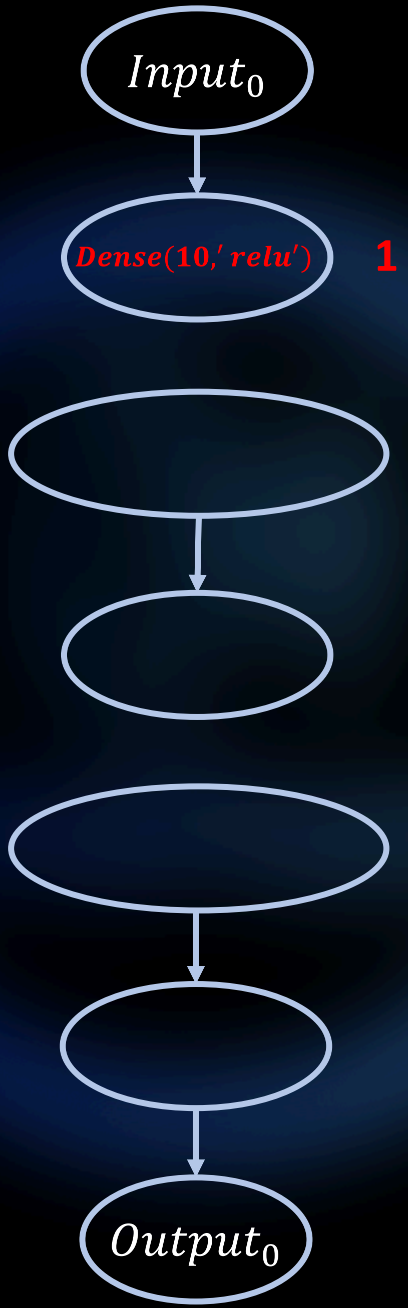
The NAS Search Space

A discrete space embedded as a directed graph where each nodes represents a choice between differents operations.



The NAS Search Space

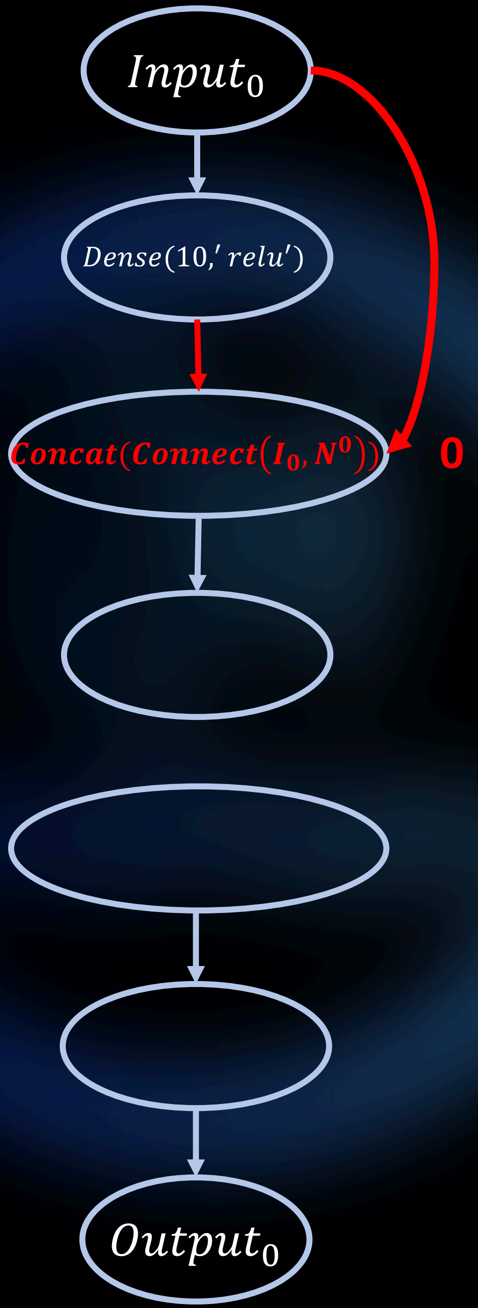
A discrete space embedded as a directed graph where each nodes represents a choice between differents operations.



Argonne NATIONAL LABORATORY

The NAS Search Space

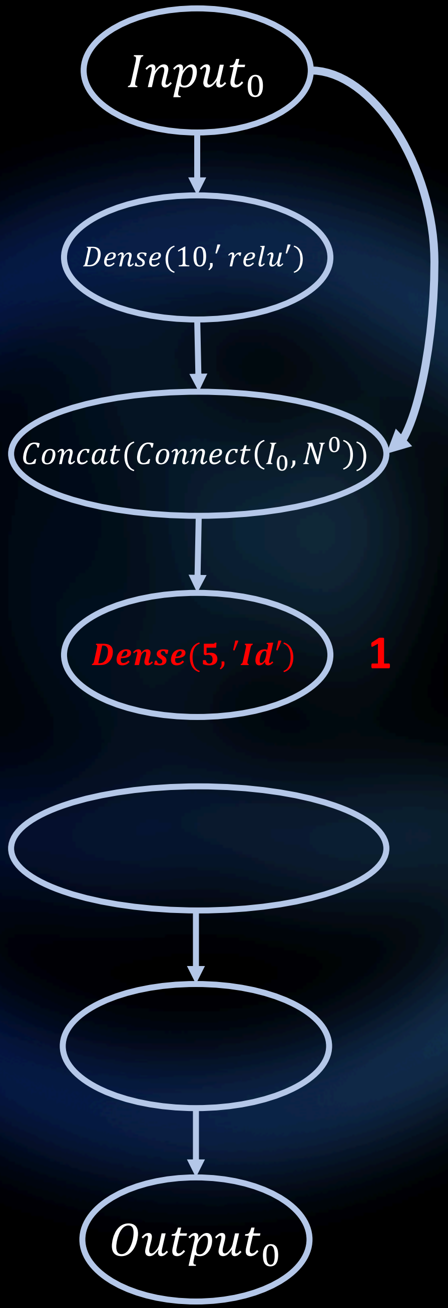
A discrete space embedded as a directed graph where each nodes represents a choice between differents operations.



Argonne NATIONAL LABORATORY

The NAS Search Space

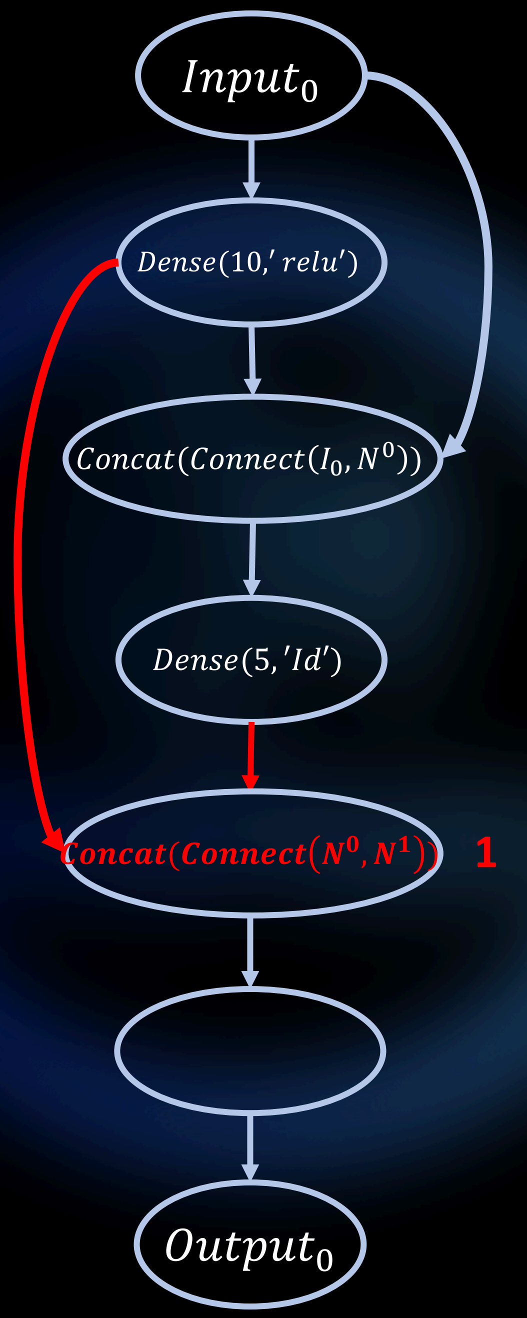
A discrete space embedded as a directed graph where each nodes represents a choice between differents operations.



Argonne NATIONAL LABORATORY

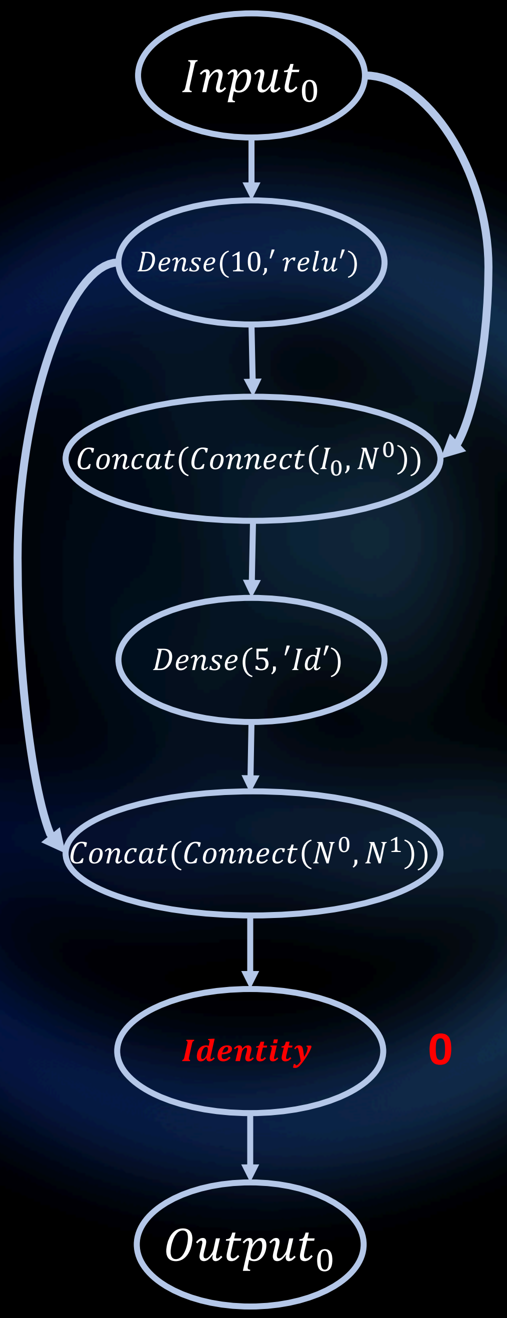
The NAS Search Space

A discrete space embedded as a directed graph where each nodes represents a choice between differents operations.



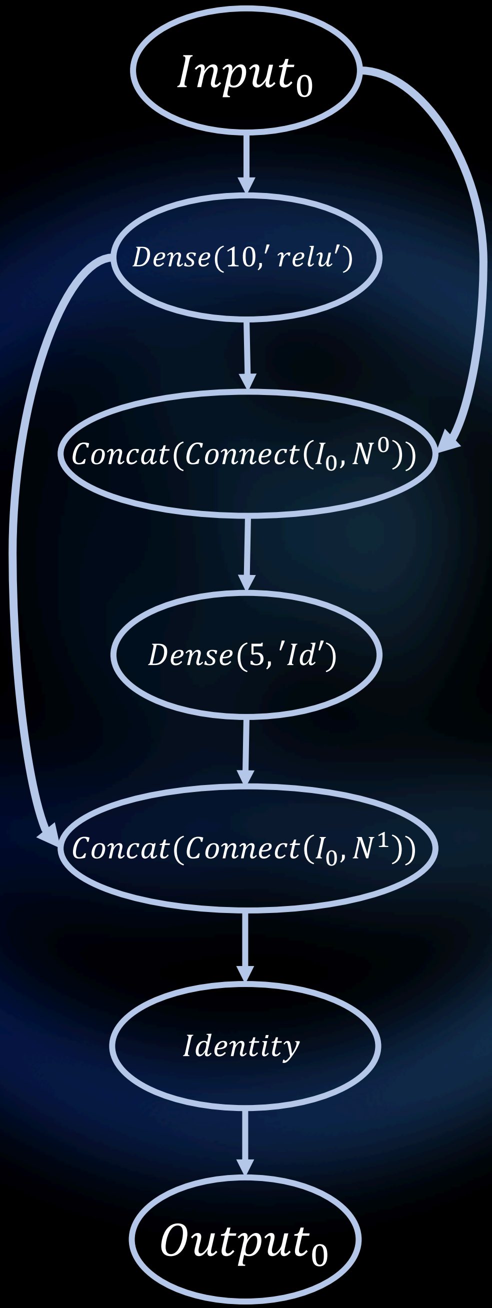
The NAS Search Space

A discrete space embedded as a directed graph where each nodes represents a choice between differents operations.



The NAS Search Space

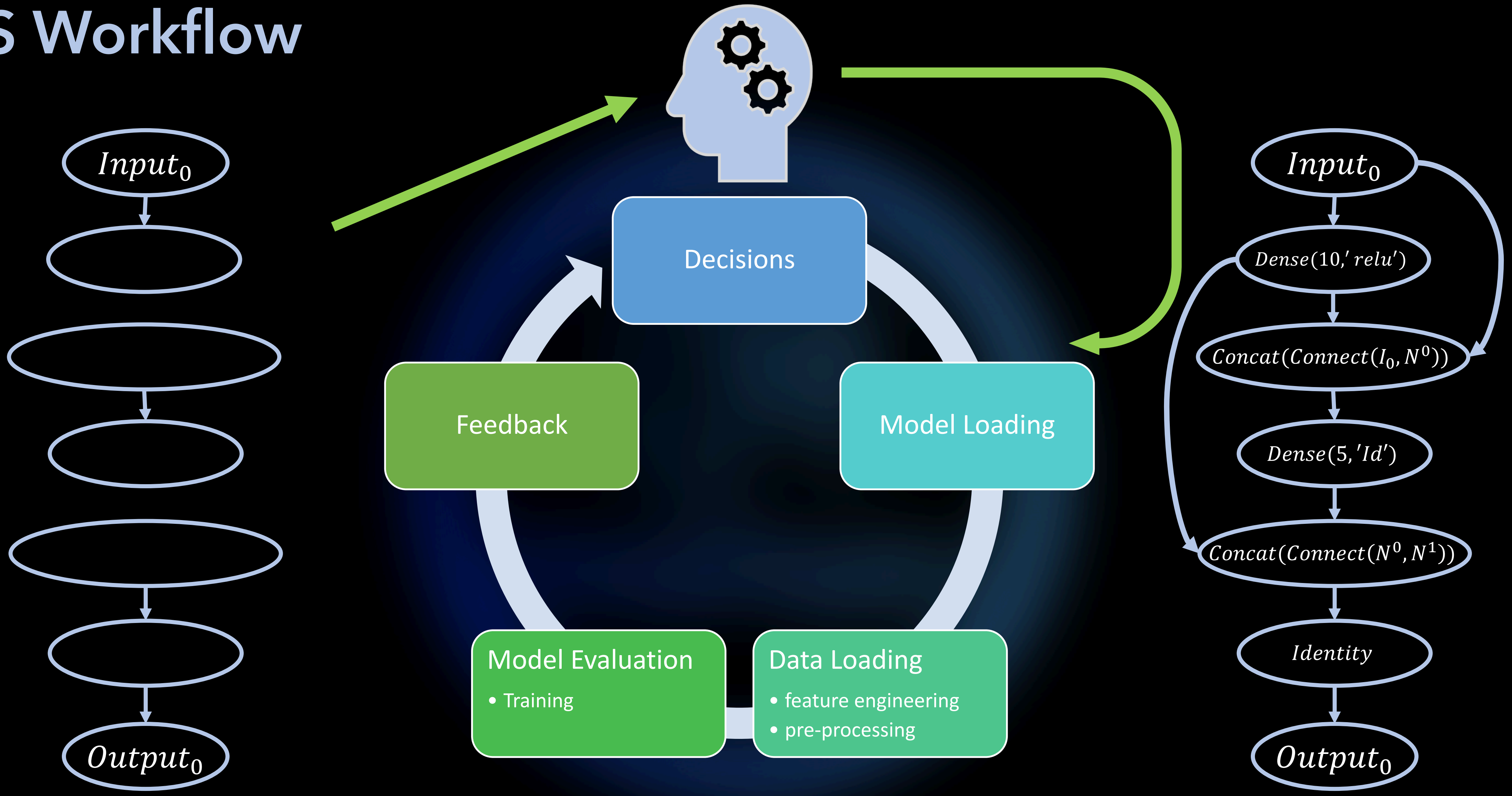
A discrete space embedded as a directed graph where each nodes represents a choice between differents operations.



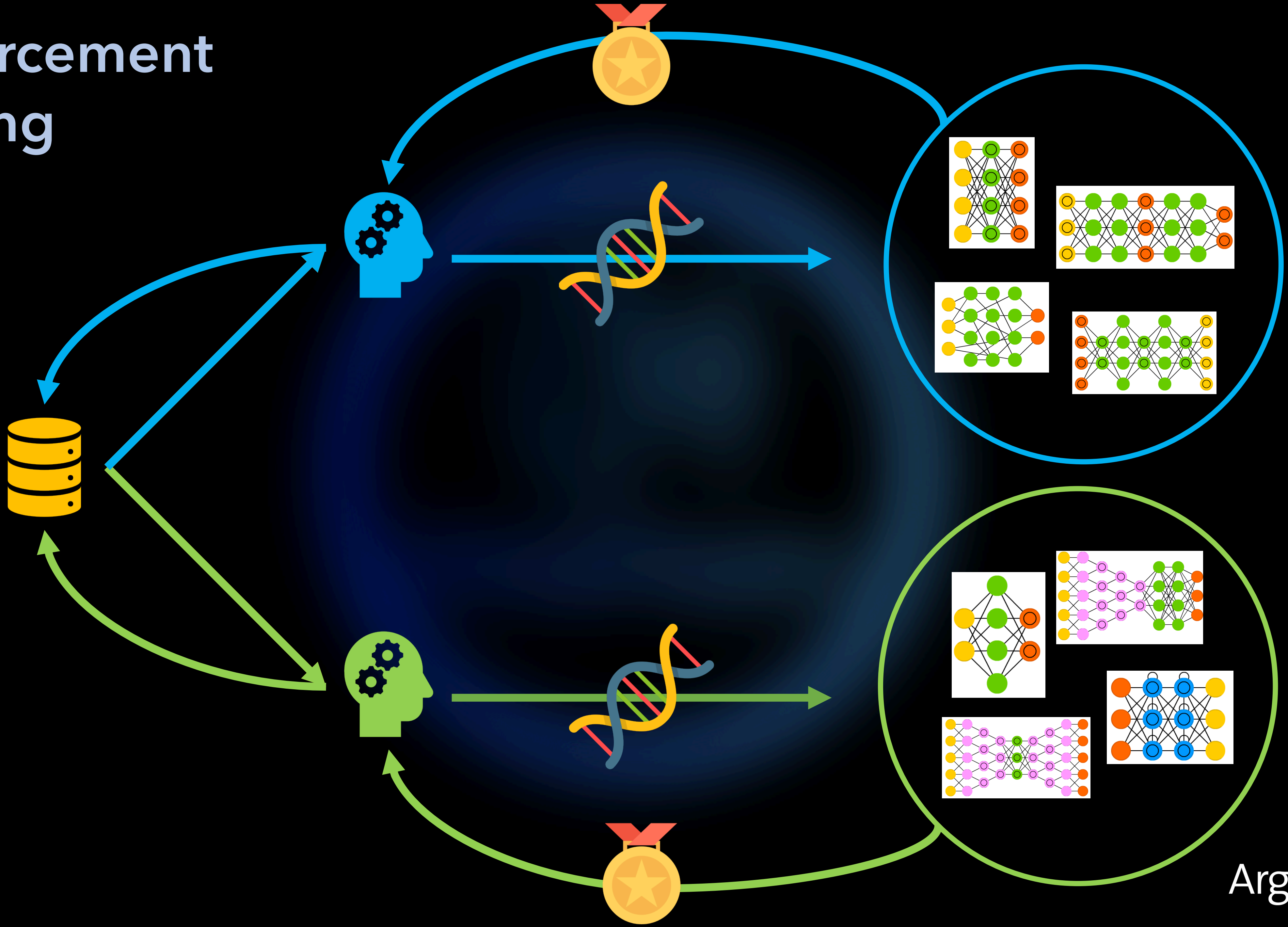
Decisions Summary

1
0
1
1
0

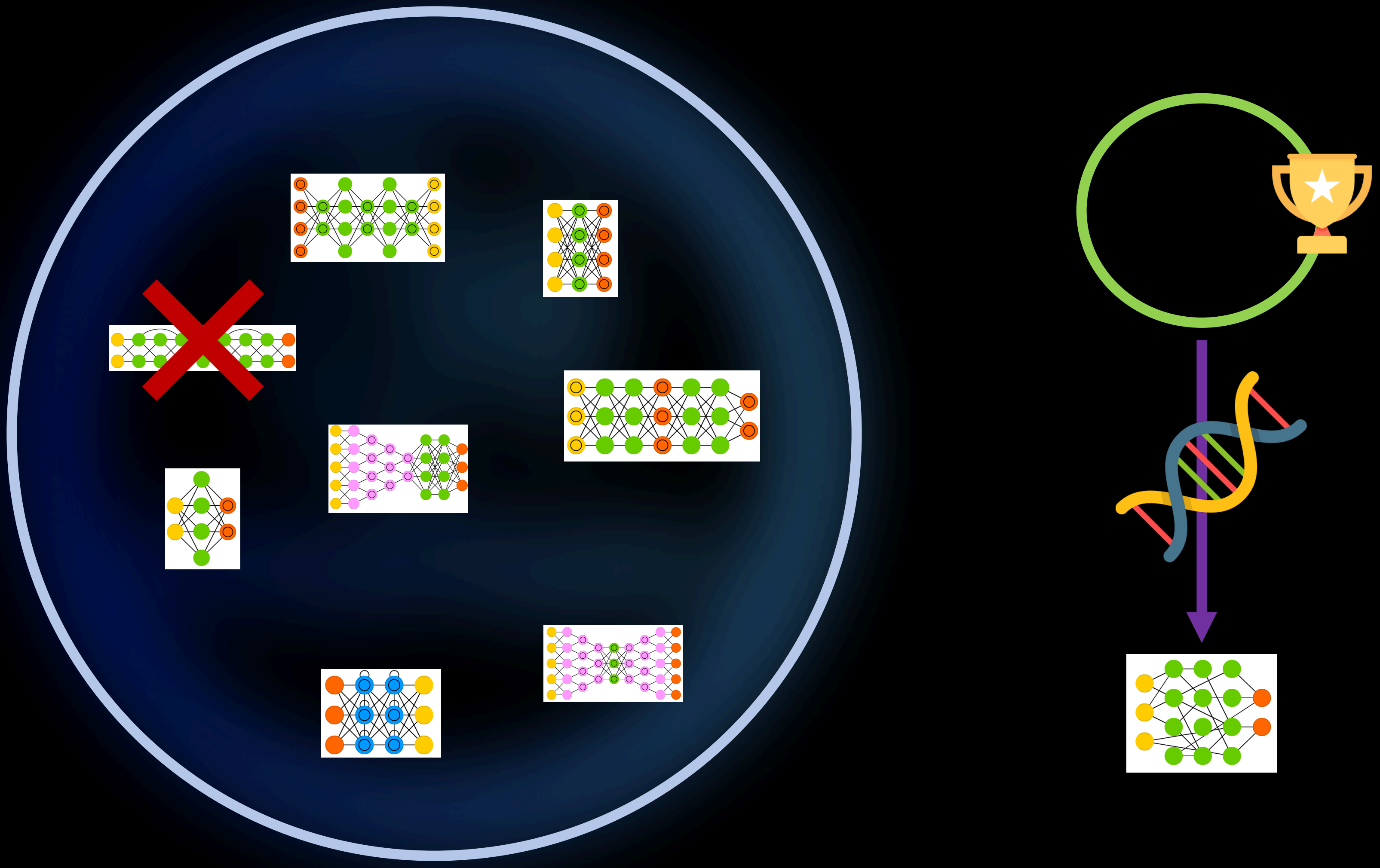
NAS Workflow



Reinforcement Learning



Aging Evolution



CANcer Distributed Learning Environment (CANDLE)

Combo

Given combination drug screening results on NCI60 cell lines predict the growth percentage from the cell line molecular features and the descriptors of both drugs.

Uno

Predict tumor dose response across multiple data sources.

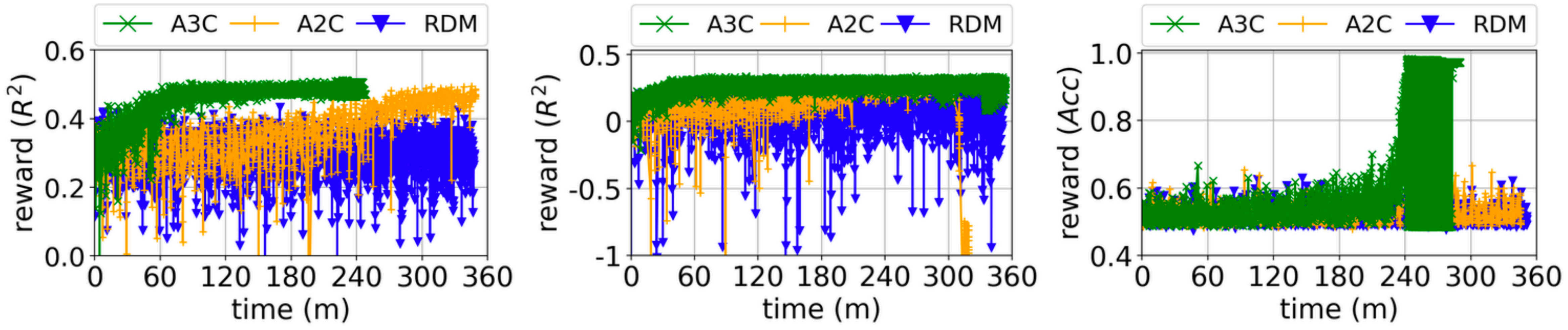
NT3

Classify RNA-seq gene expression profiles into normal or tumor tissue categories.

Argonne NATIONAL LABORATORY

CANcer Distributed Learning Environment (CANDLE)

Combo Uno NT3



(a) Combo (b) Uno (c) NT3

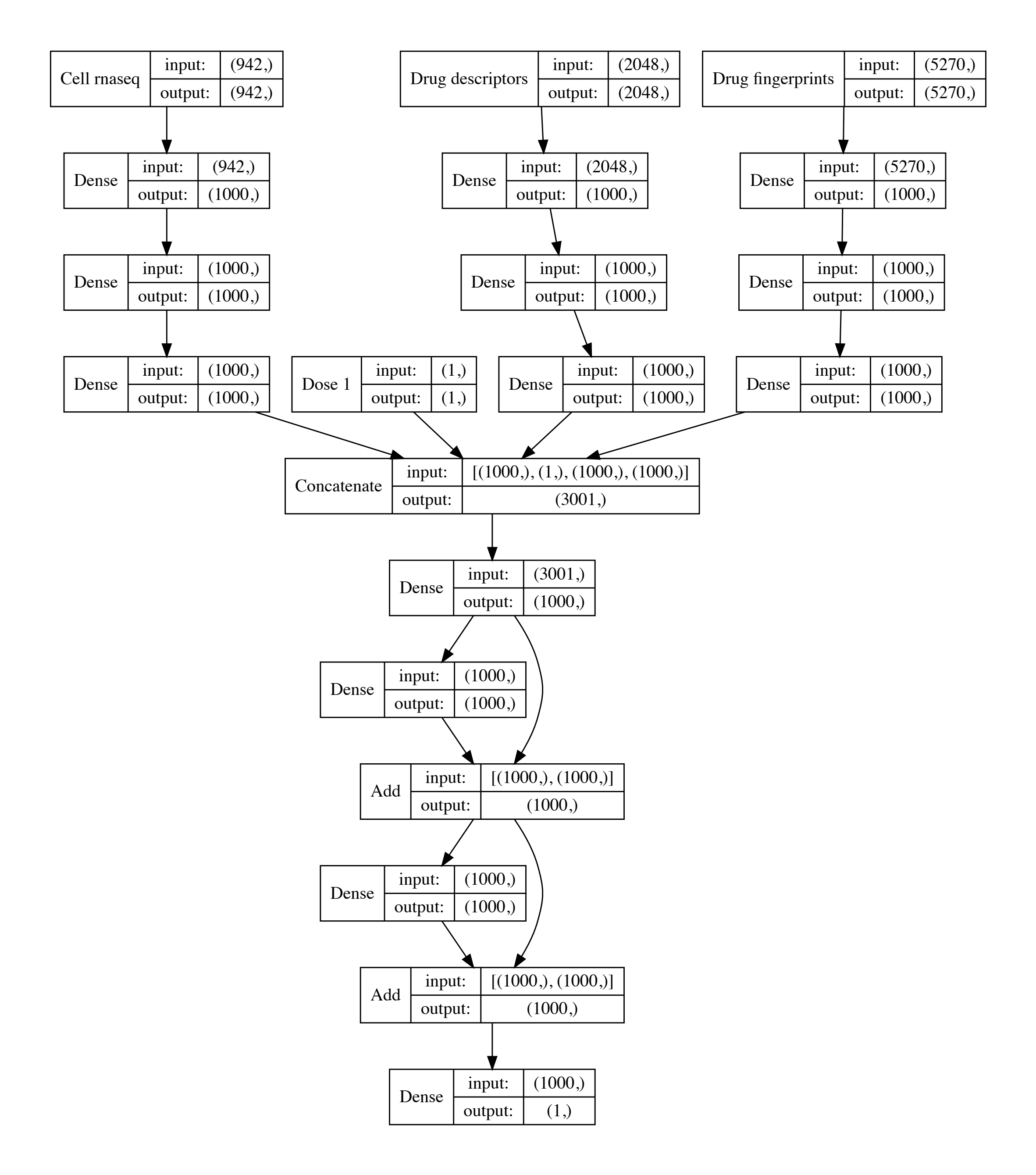
Figure 4: Search trajectory showing reward over time for A3C, A2C, and RDM on the small search space

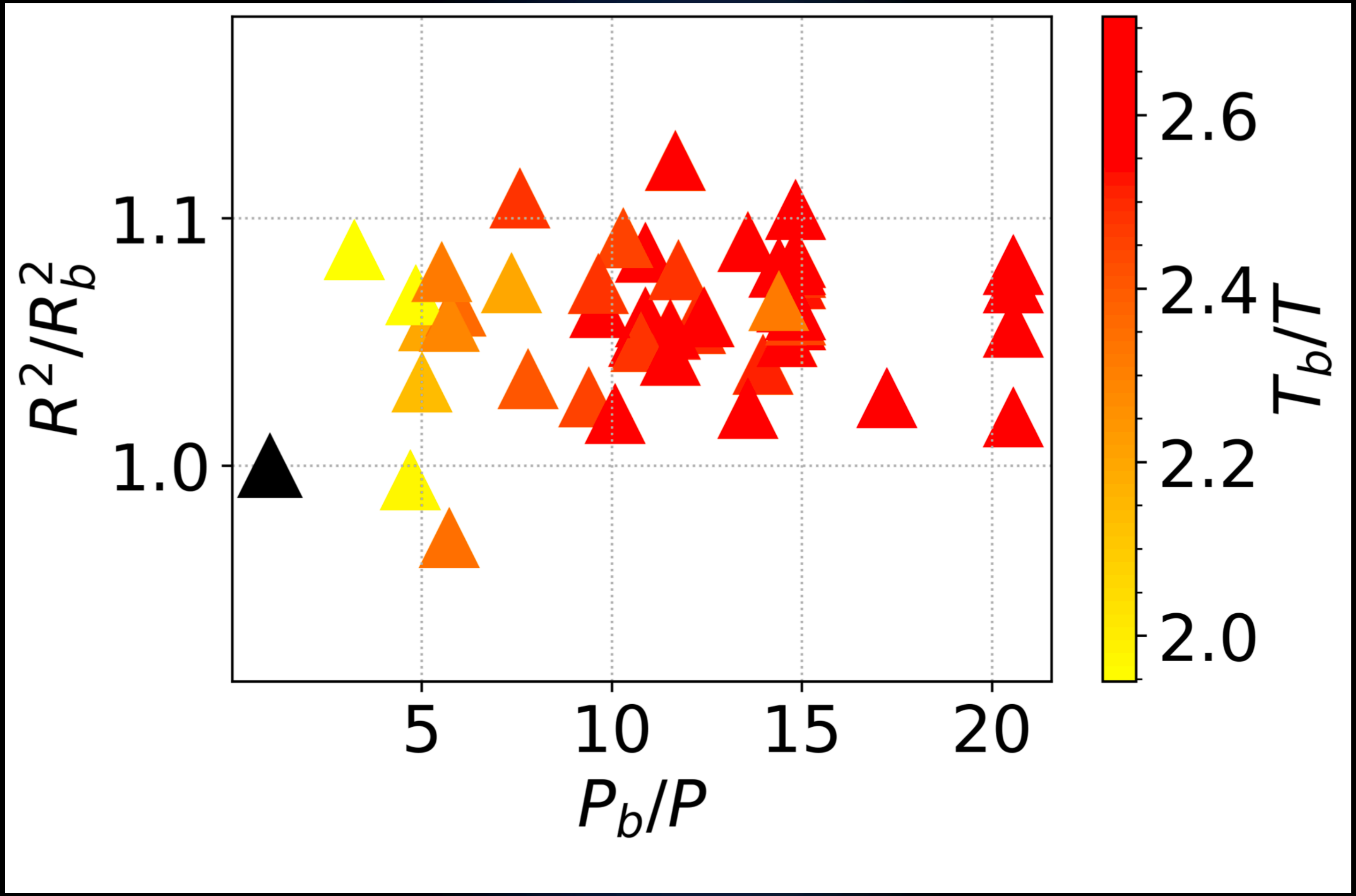
Argonne NATIONAL LABORATORY

Uno

Prédire la réponse de la tumeur au dosage d'une molécule/médicament (eng: drug) à partir de plusieurs données:

- Séquence ARN
- Dosage de la molécule
- Descripteur de la molécule
- Empreinte de la molécule





1.1
1.0
5
10
15
20
R^2/R_b^2
P_b/P
T_b/T
2.6
2.4
2.2
2.0

Best models found

	Trainable Parameters	Training Time (s)	R^2 or ACC
	Combo		
manually designed	13,772,001	705.26	0.926
A3C-best	**1,883,301**	**283.00**	**0.93**
	Uno		
manually designed	19,274,001	164.94	0.649
A3C-best	**1,670,401**	**63.53**	**0.729**
	NT3		
manually designed	96,777,878	247.63	0.986
A3C-best	**120,968**	**16.65**	**0.989**

281474976710656 possibilities

6.541611696775362e-09% explored

18413 unique models

256 nodes

6 hours

~50 selected

Argonne NATIONAL LABORATORY

Acknowledgements

DOE Early Career Research Program, ASCR

Argonne Leadership Computing Facility

Laboratory Directed Research and Development (LDRD)

Thank you!

SCAN ME

https://deephyper.readthedocs.io