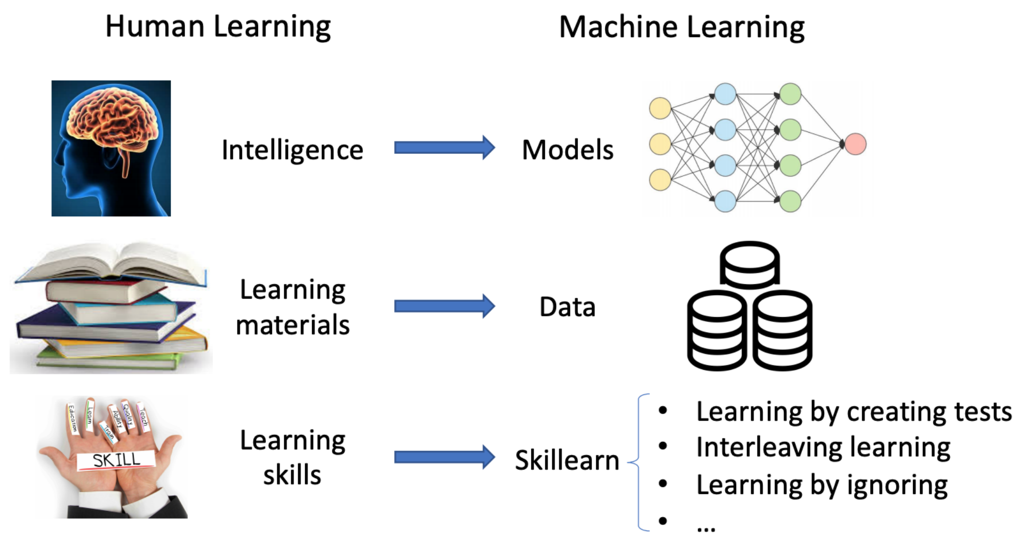
Skillearn: Machine Learning Inspired by Humans' Learning Skills

Skillearn

Problem Statement

The researchers are interested in formalizing human learning skills (learning by passing tests and interleaving learning) to train better machine learning (ML) models.

Proposed Solution



Learning by passing tests
Involves the following steps
1. Involves 2 learners, a teacher and student.
2. The student learns a target task.
3. The teacher learns to create tests in order to evaluate the student.
4. The student is able to identify his/her weaknesses and improve upon them.

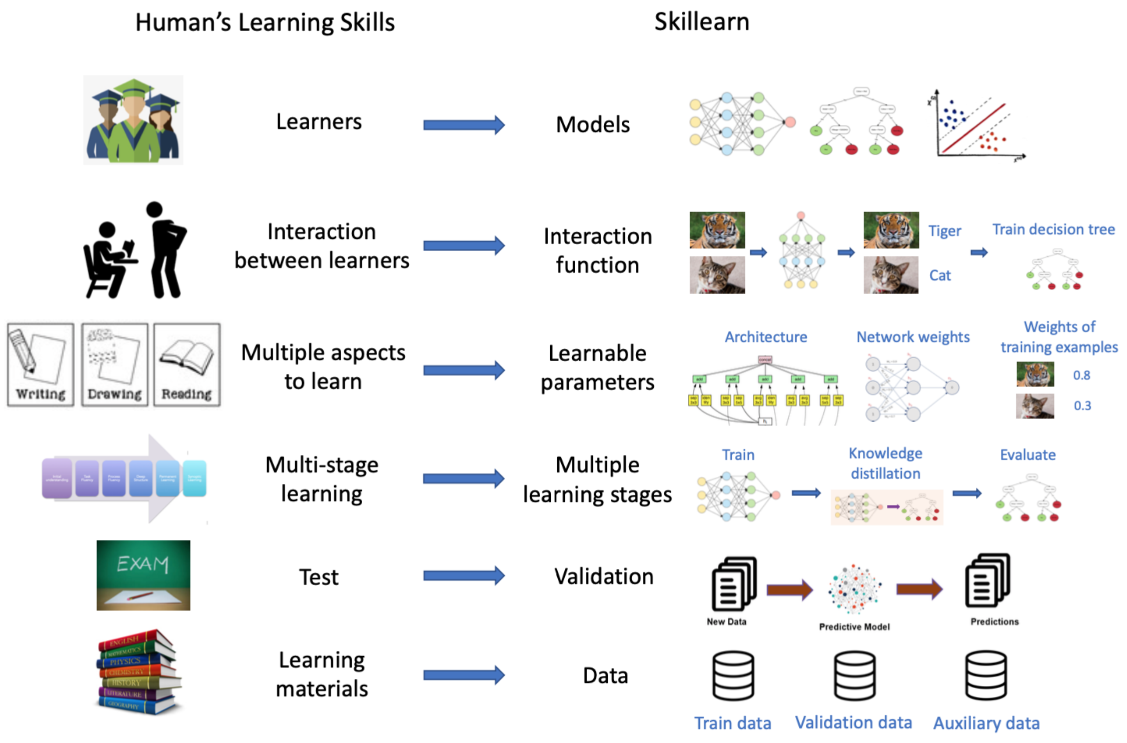
Interleaving learning
1. The learner switches between topics A,B and C.

2. He/She Studies A for a while and then switches to B and so on. Study pattern followed, ABC, ABC…
3. Compared to block learning, interleaving learning leads to higher retention and improves ability to transfer.

Basically the researchers have tried to map human learning skills (HLS) with machine learning skills.

Human learning skills

2.2. General Framework of Skillearn

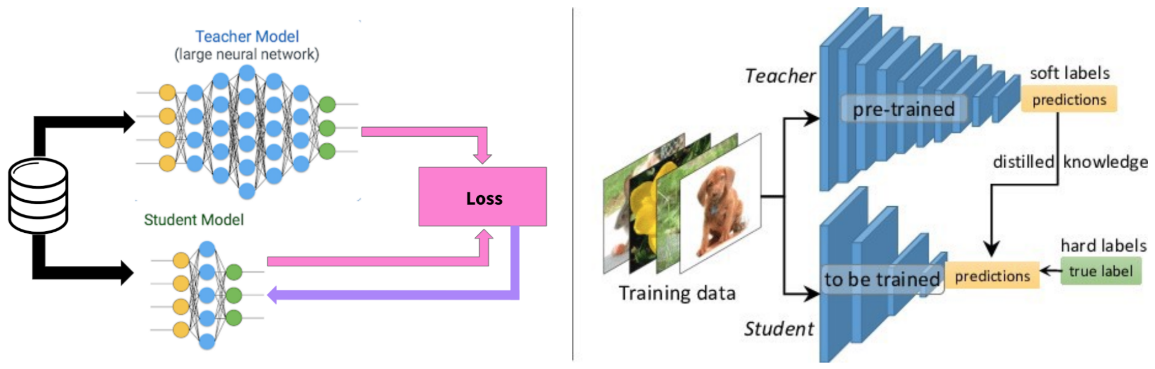


Learners

Humans: Students
ML: ML models like Deep neural networks and Convolutional neural networks.

Interaction between learners

Humans: Teacher conducts tests and evaluates them.
ML: For example, in Knowledge distillation, A ‘teacher’ model teaches a ‘student’ model by predicting pseudo labels and uses them to train the student model.



Multiple aspects to learn
Humans: Humans learn different aspects associated with learning like reading, drawing, etc.
ML: network weights, hyper-parameters, network architecture need to be learned during training .

Multi stage learning
Humans: learning by conducting tests.
ML: Example, in Knowledge distillation there are two stages
1. A teacher model is trained
2. The teacher model predicts pseudo labels on an unlabeled dataset and the pseudo-labeled dataset is used to train the student model.

Test
Humans: learning by conducting tests.
ML: We perform validation and testing of models to evaluate their robustness.

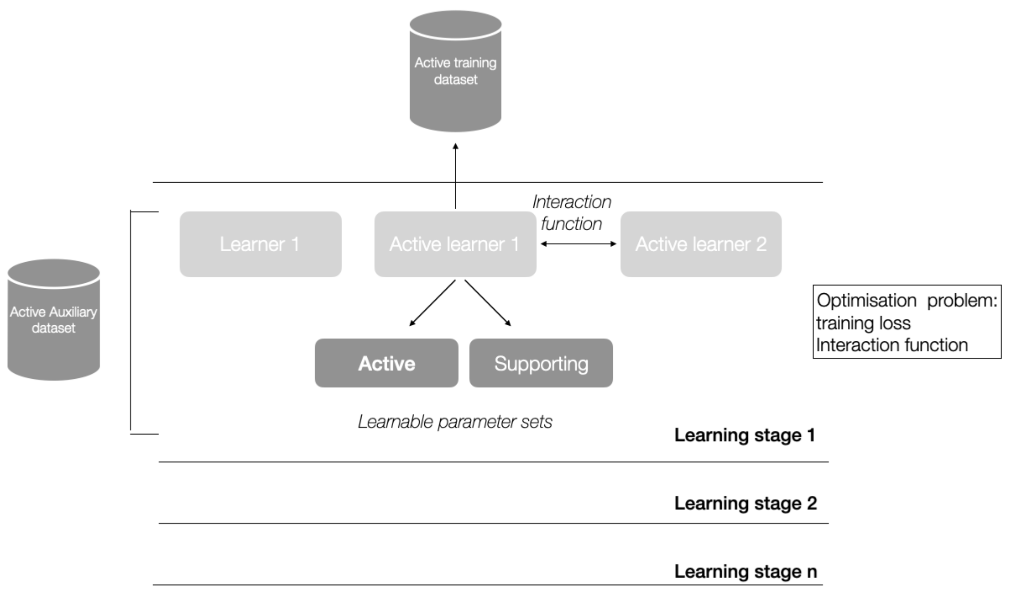
Learning materials
Humans: Books, tests, supplementary material
ML: Training, testing and auxiliary (additional labeled datasets for validation) datasets.

General framework

Learning is performed in stages. The elements of Skillearn are shown in the diagram below.

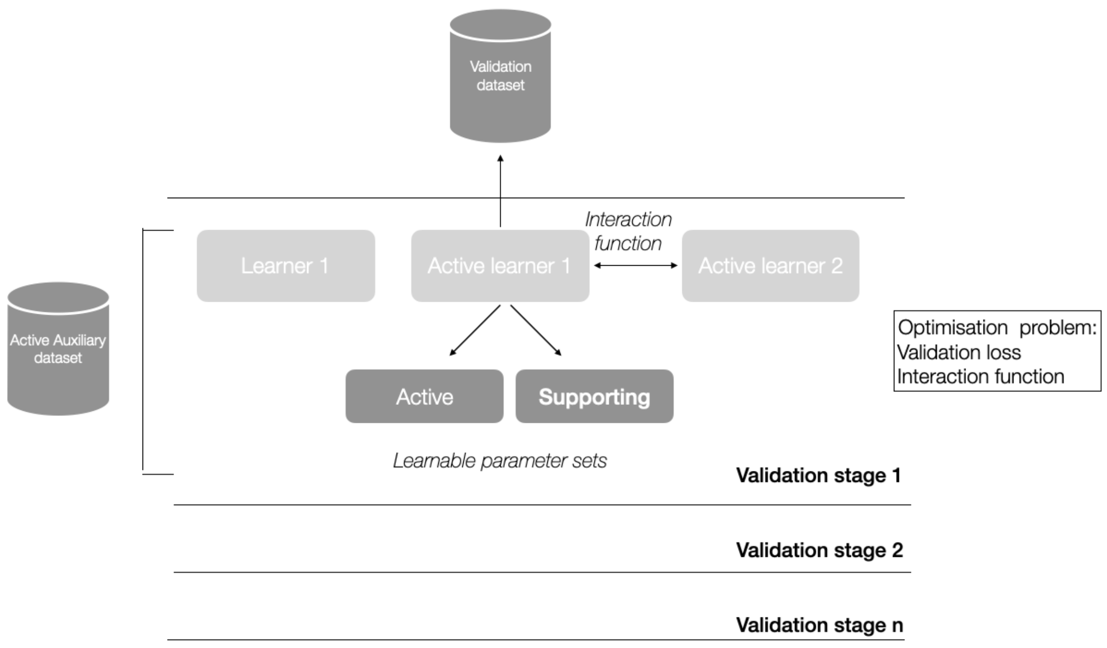
Training

The active learners and the active learning parameters are updated at this stage.

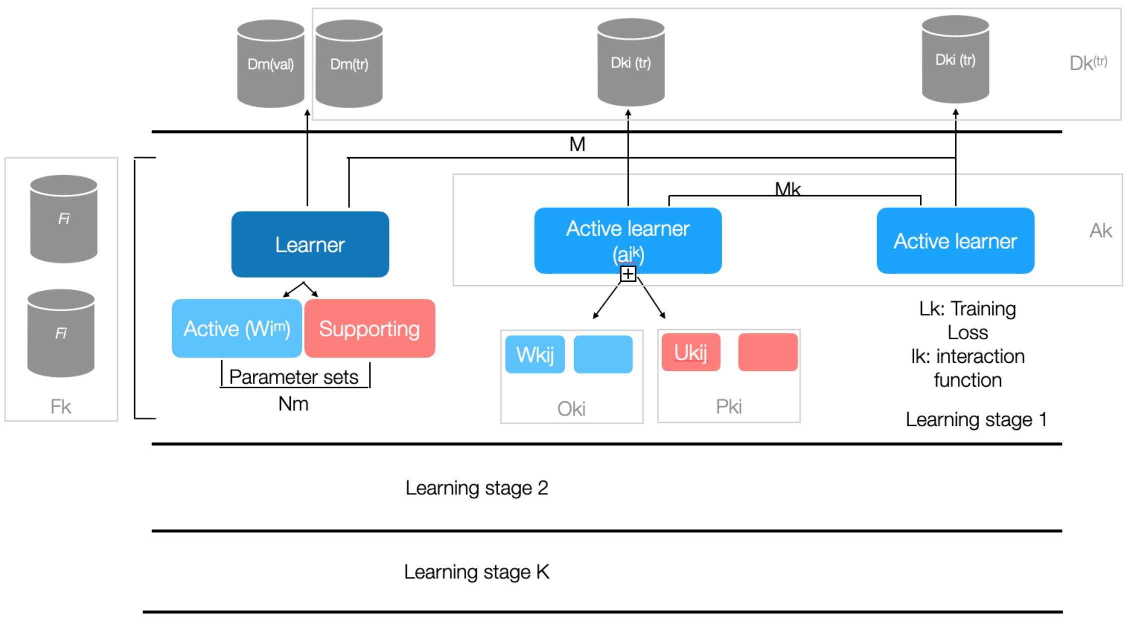


Validation

The supporting parameters (remaining parameters) are updated in the validation stage.



Mathematical Setup



Wk: Active parameter sets in the kth stage
Uk: Supporting parameter sets in the kth stage

Mathematical Framework

Training stage
The motto is to minimise the objective function. For the first learning stage, the objective function will be

Learning stage 1:

$$\underbrace{\mathcal{W}_1^*(\mathcal{U}_1)}_{\text{Objective Function}} = \min_{\mathcal{W}_1} \underbrace{L_1(\mathcal{W}_1, \mathcal{U}_1, \mathcal{D}_1^{(\mathrm{tr})}, \mathcal{F}_1)}_{\text{Training Loss}} + \gamma_1 \underbrace{I_1(\mathcal{W}_1, \mathcal{U}_1, \mathcal{F}_1)}_{\text{Interaction function}}$$

Where training loss is a function of,
$D1^{(tr)}$- set of Active training datasets
W1- set of Active learnable parameters
U1- set of Supporting learnable parameters
F1- set of Auxiliary datasets (optional)

Interaction function depends on,
W1- set of Active learnable parameters
U1- set of Supporting learnable parameters
F1- set of Auxiliary datasets
And ***γ***1 is a tradeoff parameter between the training loss and the interaction function.

Similarly for k stages,

$$
\begin{aligned}
s.t. \quad & \text{Learning stage } K\text{:} \\
& \mathcal{W}_K^*(\{\mathcal{U}_j\}_{j=1}^{K}) = \min_{\mathcal{W}_K} L_K(\mathcal{W}_K, \mathcal{U}_K, \{\mathcal{W}_j^*(\{\mathcal{U}_i\}_{i=1}^{j})\}_{j=1}^{K-1}, \mathcal{D}_K^{(\mathrm{tr})}, \mathcal{F}_K) + \\
& \qquad\qquad \gamma_K I_K(\mathcal{W}_K, \mathcal{U}_K, \{\mathcal{W}_j^*(\{\mathcal{U}_i\}_{i=1}^{j})\}_{j=1}^{K-1}, \mathcal{F}_K) \\
& \vdots \\
& \text{Learning stage } k\text{:} \\
& \mathcal{W}_k^*(\{\mathcal{U}_j\}_{j=1}^{k}) = \min_{\mathcal{W}_k} L_k(\mathcal{W}_k, \mathcal{U}_k, \{\mathcal{W}_j^*(\{\mathcal{U}_i\}_{i=1}^{j})\}_{j=1}^{k-1}, \mathcal{D}_k^{(\mathrm{tr})}, \mathcal{F}_k) + \\
& \qquad\qquad \gamma_k I_k(\mathcal{W}_k, \mathcal{U}_k, \{\mathcal{W}_j^*(\{\mathcal{U}_i\}_{i=1}^{j})\}_{j=1}^{k-1}, \mathcal{F}_k) \\
& \vdots \\
& \text{Learning stage 1:} \\
& \mathcal{W}_1^*(\mathcal{U}_1) = \min_{\mathcal{W}_1} L_1(\mathcal{W}_1, \mathcal{U}_1, \mathcal{D}_1^{(\mathrm{tr})}, \mathcal{F}_1) + \gamma_1 I_1(\mathcal{W}_1, \mathcal{U}_1, \mathcal{F}_1)
\end{aligned}
\tag{1}
$$

For the k th stage, the training loss(Lk) is dependent upon,
Wk- Set of active learnable parameters for the kth stage
Uk- Set of supporting learnable parameters for the kth stage
$W^*j(\{Ui\}_{i=1}^{j})^{k-1}{}_{j=1}$- Optimal solutions obtained in previous learning stages.
Dk- Set of active training datasets for the kth stage
Fk- Set of active auxiliary datasets for the kth stage

And the Interaction function is dependent upon,
Wk- Set of Active learnable parameters for the kth stage
Uk- Set of Supporting learnable parameters for the kth stage
$W^*j(\{Ui\}_{i=1}^{j})^{k-1}{}_{j=1}$-Optimal solutions obtained in previous learning stages.
Fk- Set of active auxiliary datasets for the kth stage

Note:

1. W*k is a function of $(\{Ui\}_{i=1}^{j})^{k}{}_{j=1}$ as W*k is a function of the objective and the objective inturn is a function of $(\{Ui\}_{i=1}^{j})^{k}{}_{j=1}$.
2. The active learners do not interact in the training loss.

Validation stage

In the validation stage we train the supporting learning parameters that were left out in the training process. We will be using the validation sets for all the learners in this stage.

$$
\max_{\{\mathcal{U}_i\}_{i=1}^{K}} \quad L_{val}(\{\mathcal{W}_j^*(\{\mathcal{U}_i\}_{i=1}^{j})\}_{j=1}^{K}, \mathcal{D}^{(\mathrm{val})}, \mathcal{F}) + \gamma_{val} I_{val}(\{\mathcal{W}_j^*(\{\mathcal{U}_i\}_{i=1}^{j})\}_{j=1}^{K}, \mathcal{F}) \quad \text{(Validation stage)}
$$

Optimization Algorithm

Training stage

Liu et al have proposed a simple approximation scheme that omits solving the optimization completely by training until convergence. The idea is to approximate W*k({Uj}$_{j=1}^{k}$) by adapting W using a single training step. Hence the approximated optimal solution after one step gradient descent update can be written as,

$$\mathcal{W}_k^*(\{\mathcal{U}_j\}_{j=1}^k) = \min_{\mathcal{W}_k} L_k(\mathcal{W}_k, \mathcal{U}_k, \{\mathcal{W}_j^*(\{\mathcal{U}_i\}_{i=1}^j)\}_{j=1}^{k-1}, \mathcal{D}_k^{(\mathrm{tr})}, \mathcal{F}_k) + \gamma_k I_k(\mathcal{W}_k, \mathcal{U}_k, \{\mathcal{W}_j^*(\{\mathcal{U}_i\}_{i=1}^j)\}_{j=1}^{k-1}, \mathcal{F}_k)$$

$$\begin{aligned} &\mathcal{W}_k^*(\{\mathcal{U}_j\}_{j=1}^k) \approx \mathcal{W}_k'(\{\mathcal{U}_j\}_{j=1}^k) = \\ &\mathcal{W}_k - \eta \nabla_{\mathcal{W}_k}(L_k(\mathcal{W}_k, \mathcal{U}_k, \{\mathcal{W}_j^*(\{\mathcal{U}_i\}_{i=1}^j)\}_{j=1}^{k-1}, \mathcal{D}_k^{(\mathrm{tr})}, \mathcal{F}_k) + \gamma_k I_k(\mathcal{W}_k, \mathcal{U}_k, \{\mathcal{W}_j^*(\{\mathcal{U}_i\}_{i=1}^j)\}_{j=1}^{k-1}, \mathcal{F}_k)). \end{aligned} \tag{3}$$

Where,
W'k({Ui}$_{j=1}^{k}$)- Approximated objective

Validation Stage

Similarly the objective for the validation stage can be written as,

$$L_{val}(\{\mathcal{W}_j'(\{\mathcal{U}_i\}_{i=1}^j)\}_{j=1}^K, \mathcal{D}^{(\mathrm{val})}, \mathcal{F}) + \gamma_{val} I_{val}(\{\mathcal{W}_j'(\{\mathcal{U}_i\}_{i=1}^j)\}_{j=1}^K, \mathcal{F}). \tag{5}$$

The remaining learning parameters ({Ui}$_{i=1}^{j}$) are estimated by minimizing this objective.

Optimisation Algorithm

Algorithm 1 Optimization algorithm for Skillearn

```
while not converged do
    1. For $k = 1 \cdots K$, update $\mathcal{W}_k^*(\{\mathcal{U}_j\}_{j=1}^k)$ using Eq.(4)
    2. Update $\{\mathcal{U}_i\}_{i=1}^K$ by minimizing the approximated objective in Eq.(5)
end
```

Learning by Passing Tests. (LPT)

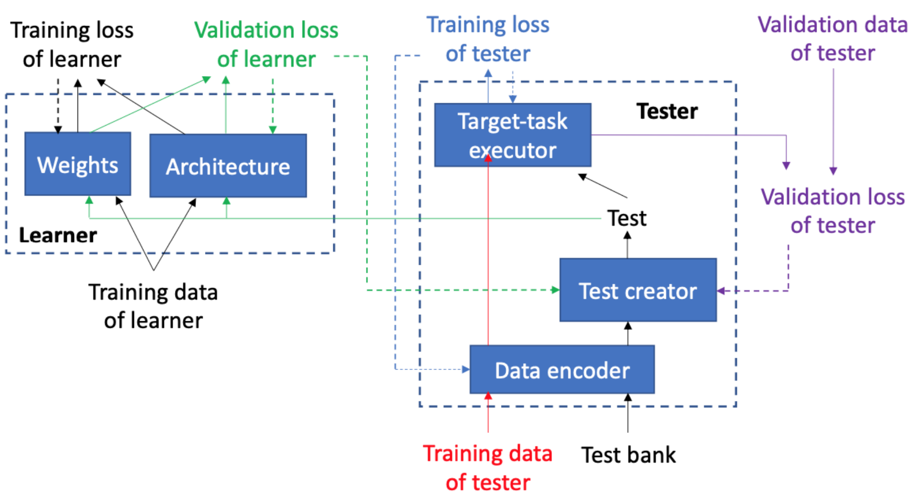
Here we apply the generalised framework proposed above to train a machine learning model by the LPT methodology.
This framework consists of 2 learners,
Testee model - It tries to learn better so that it can pass the tests.
Tester model - It creates tests with growing levels of difficulty.

The testing procedure can be described by the following algorithm

1. R ← threshold value of error
2. Tester selects test ‘T’ from test bank
3. Testee gives test, i.e applied model ‘M’
4. Error rate → R’
5. If R’ < R
    a. Creates a more difficult test T’ (T ← T’)
    b. Go to step 3
6. Else
    a. Testee relearns M, M’ → improved model
    b. Go to step 3

This process goes on until an equilibrium is reached.

LPT framework



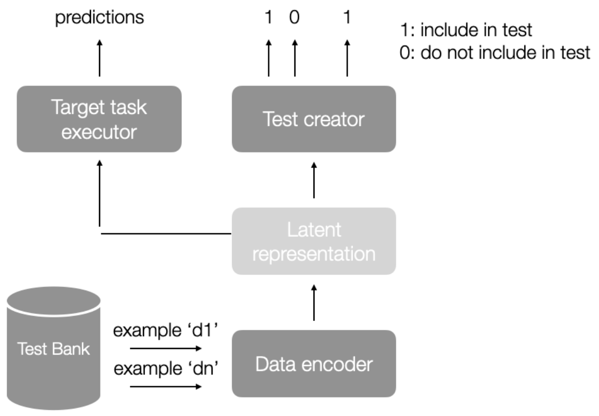
The solid arrows denote the process of making predictions and calculating losses. The dotted arrows denote the process of updating learnable parameters by minimizing corresponding losses.

The LPT framework consists of 2 entities,

Learner
Training stage:
1. The training data of the learner is used to modify the weights and architecture of the learner by reducing the training loss.
2. It is trained on the target task J1.

Validation stage:
1. The learner evaluates itself by taking the test created by the tester.
2. It sends its validation loss to the test creator so that it can improve the quality of tests.

Tester
It has three modules with learnable parameters.
a. Target task executor: Performs the task of conducting target task J2.
b. Test creator: Responsible for creating difficult and meaningful tests.
c. Data encoder:



Training stage:
1. The training data of the tester is used to train the data encoder and target task executor. This ensures that the tests are difficult for the testee/learner.
2. The test creator creates tests that are used to train the target task executor and data encoder on a target task J2 (Eg: classification). High performance of the tester ensures that the tests are meaningful.

Validation stage:

1. The tester improves its test creator from the loss obtained from the testee and by minimizing the losses on the validation set of the tester.

Mathematical framework

Similar to the general framework, we define the notations used.

Notation	Meaning
A	Architecture of the testee
W	Network weights of the testee
E	Data encoder of the tester
C	Test creator of the tester
X	Target-task executor of the tester
$D_{ee}^{(\mathrm{tr})}$	Training data of the testee
$D_{er}^{(\mathrm{tr})}$	Training data of the tester
$D_{er}^{(\mathrm{val})}$	Validation data of the tester
D_b	Test bank

Table 2: Notations in Learning by Passing Tests

Learning stages

Stage 1

The learner updates its weights using its training dataset to perform target task J1.

Active learners	Testee
Active learnable parameters	Network weights of the testee
Supporting learnable parameters	Architecture of the testee
Active training datasets	Training dataset of target-task J_1 performed by the testee
Active auxiliary datasets	–
Training loss	Training loss of target-task J_1: $L(A, W, D_{ee}^{(\mathrm{tr})})$
Interaction function	–
Optimization problem	$W^*(A) = \min_W L(A, W, D_{ee}^{(\mathrm{tr})})$

Table 3: Learning Stage I in LPT

Optimisation problem: Reduce training loss dependent on the architecture, network weights and training data.
Note: If A is learned by minimizing this training loss, a trivial solution will be yielded where A is very large and complex that it can perfectly overfit the training data but will generalize poorly on unseen data.

Stage 2
The tester learns its data encoder (E) and target task executor (X) by minimizing the training loss for task J2. Since the tester uses both its training set and the created tests to train its data encoder and target task executor, the training loss comprises 2 parts.

$$\underbrace{E^*(C), X^*(C)}_{\text{Optimally trained Encoder and test executor}} = \min_{E,X} \; \underbrace{L\left(E, X, D_{er}^{(\mathrm{tr})}\right)}_{\text{Defined on training dataset of J2}} + \underbrace{\overset{\text{Tradeoff parameter}}{\gamma} L\left(E, X, \overbrace{\sigma\left(C, E, D_b\right)}^{\text{Collection of test samples}}\right)}_{\text{Defined on test σ(C,E,Db) created by tester}}. \tag{7}$$

Active learners	Examiner
Active learnable parameters	1) Data encoder of the tester; 2) Target-task executor of the tester.
Supporting learnable parameters	Test creator of the tester
Active training datasets	Training data of target-task J_2 performed by the tester
Active auxiliary datasets	Test bank
Training loss	$L(E, X, D_{er}^{(\mathrm{tr})}) + \gamma L(E, X, \sigma(C, E, D_b))$
Interaction function	–
Optimization problem	$E^*(C), X^*(C) = \min_{E,X} L(E, X, D_{er}^{(\mathrm{tr})}) + \gamma L(E, X, \sigma(C, E, D_b))$.

Table 4: Learning Stage II in LPT

Optimisation problem: Reduce the loss defined on the training set and the loss defined on the test.

Note:

1. Creator C is used to define the loss, but it is not learned during the training stage, as a trivial solution would be setting all values to ‘0’ for all test bank samples.
2. Again, E and X are functions of the training loss and the training loss is a function of C.

Stage 3

Testee updates its architecture by minimising the validation loss to pass the test σ(C,E*(C),Db) created by the tester.

$$\underbrace{L(A, W^*(A), \sigma(C, E^*(C), D_b))}_{\substack{\text{predictive loss of testee on} \\ \text{test σ(C,E*(C),Db) on test samples} \\ \text{Db}}} = \underbrace{\sum_{d \in \sigma(C, E^*(C), D_b)} \ell(A, W^*(A), d)}_{\substack{\text{Summation of loss for} \\ \text{each test sample d} \\ \text{of test σ(C,E*(C),Db)}}} \tag{8}$$

A smaller predictive loss indicates the learner/testee is performing well in the test. On the other hand, smaller predictive loss is an indication to the tester to improve its tests. Hence the learner’s aim is to reduce the predictive loss while the tester’s aim is to increase it in order to produce more robust tests. Essentially the tester and testee interact in an adversarial manner.

Note:

1. A trivial solution for the tester to increase the predictive loss is by increasing the test samples, but more test samples do not indicate a more difficult test. Hence we normalise the loss,

$$\frac{1}{\underbrace{|\sigma(C, E^*(C), D_b)|}_{\substack{\text{cardinality of} \\ \text{set σ(C,E*(C),Db)}}}} L(A, W^*(A), \sigma(C, E^*(C), D_b)) \tag{9}$$

Validation stage

1. The testee updates its architecture to minimize the validation loss on the tests.
2. The tester updates its test creator by maximising the learner's prediction loss and decreasing its own validation loss.

Optimisation problem

$$\max_{C}\min_{A} \underbrace{\frac{1}{|\sigma\left(C, E^{*}(C), D_{b}\right)|} L\left(A, W^{*}\left(A\right), \sigma\left(C, E^{*}(C), D_{b}\right)\right)}_{\text{maximise validation loss of learner}} - \lambda L\left(\underbrace{E^{*}(C), X^{*}(C), D_{er}^{(\mathrm{val})}}_{\text{minimize tester's validation loss}}\right). \tag{10}$$

Finally, we have 3 optimisation problems, On the constraints of the outer optimization problem are two inner optimization problems corresponding to the first and second learning stage respectively.

$$\begin{aligned}
&\max\nolimits_{C}\min\nolimits_{A} \quad \tfrac{1}{|\sigma(C,E^{*}(C),D_{b})|} L\left(A, W^{*}\left(A\right), \sigma\left(C, E^{*}(C), D_{b}\right)\right) - \lambda L\left(E^{*}(C), X^{*}(C), D_{er}^{(\mathrm{val})}\right) \text{ (III)}\\
&s.t. \quad E^{*}(C), X^{*}(C) = \min\nolimits_{E,X} \quad L\left(E, X, D_{er}^{(\mathrm{tr})}\right) + \gamma L\left(E, X, \sigma\left(C, E, D_{b}\right)\right) \text{ (Stage II)}\\
&\qquad W^{*}\left(A\right) = \min\nolimits_{W} \quad L\left(A, W, D_{ee}^{(\mathrm{tr})}\right) \text{ (Stage I)}
\end{aligned} \tag{11}$$

Figuring out the math,

The test σ(C,E*(C),Db) is a discrete set and finding the optimal set of testing samples is a computationally expensive task. Similar to Liu et al, to make the search space continuous, the researchers perform relaxation over σ(C,E*(C),Db). Using relaxation we approximate the solution to the problem of searching over a discrete set.

$$L(E, X, \sigma(C, E, D_{b})) = \underbrace{\sum_{d \in D_{b}} f(d, C, E)}_{\text{Weighting parameter}} \ell(E, X, d) \tag{13}$$

Similarly,

$$L(A, W^*(A), \sigma(C, E^*(C), D_b)) \quad = \quad \textstyle\sum_{d \in D_b} f(d, C, E^*(C)) \ell(A, W^*(A), d)$$

$$|\sigma(C, E^*(C), D_b)| = \sum_{d \in D_b} f(d, C, E^*(C)) \tag{14}$$

Hence, we are calculating individual losses for each test sample d and summing it up.

Optimization

Stage 1
We approximate W*(A) by one step gradient descent method.

$$\begin{aligned} & \nabla_A L\left(A, W^*(A), \sigma\right) \approx \\ & \nabla_A L\left(A, W-\xi_{ee} \nabla_W L\left(A, W, D_{ee}^{(\mathrm{tr})}\right), \sigma\right)= \\ & \nabla_A L\left(A, W^{\prime}, \sigma\right)-\xi_{ee} \nabla_{A, W}^2 L\left(A, W, D_{ee}^{(\mathrm{tr})}\right) \nabla_{W^{\prime}} L\left(A, W^{\prime}, \sigma\right), \end{aligned} \tag{16}$$

Here we use the concept of Implicit and Explicit gradients to simplify $\nabla_A L(A,W^*(A),\sigma)$. This is used when a function parameter is itself a function of another parameter i.e, W is a function of A in our case.

Implicit and Explicit gradients

$$\frac{\partial \mathrm{f}(\mathrm{z},\mathrm{x})}{\partial \mathrm{x}} = \frac{\partial^+ \mathrm{f}(\mathrm{z},\mathrm{x})}{\partial \mathrm{x}} + \frac{\partial \mathrm{f}(\mathrm{z},\mathrm{x})}{\partial \mathrm{z}} \frac{\partial \mathrm{z}}{\partial \mathrm{x}}$$

where $\frac{\partial^+ \mathrm{f}(\mathrm{z},\mathrm{x})}{\partial \mathrm{x}}$ is called **implicit gradient**
and $\frac{\partial \mathrm{f}(\mathrm{z},\mathrm{x})}{\partial \mathrm{z}} \frac{\partial \mathrm{z}}{\partial \mathrm{x}}$ is called **explicit gradient**

The term $\xi_{ee} \nabla^2_{A,W} L(A,W,D_{ee}^{(tr)})$involves computationally expensive matrix product calculations which can be reduced by finite difference approximation.

Finite Difference Method

$$\frac{\partial f(x,z)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon,z) - f(x-\epsilon,z)}{2\epsilon} = \lim_{\epsilon \to 0} \frac{f(x+k\epsilon,z) - f(x-k\epsilon,z)}{2k\epsilon} \tag{2}$$

$$\frac{\partial f(x,z)}{\partial x} * k = \lim_{\epsilon \to 0} \frac{f(x+k\epsilon,z) - f(x-k\epsilon,z)}{2\epsilon} \tag{3}$$

Applying this method we can simplify (16) as

$$\nabla^2_{A,W} L\left(A, W, D^{(\mathrm{tr})}_{ee}\right) \nabla_{W'} L\left(A, W', \sigma\right) \approx \tfrac{1}{2\alpha_{ee}}\left(\nabla_A L\left(A, W^+, D^{(\mathrm{tr})}_{ee}\right) - \nabla_A L\left(A, W^-, D^{(\mathrm{tr})}_{ee}\right)\right), \tag{17}$$

Where,
$\alpha_{ee} = \varepsilon = 0.01 / /\!/ \nabla W' \ L(A,W' \ ,\sigma)) /\!/ \ 2.$
$W\pm = W \pm \alpha_{ee} \nabla W' \ L(A,W' \ ,\sigma)$
$k = \nabla W' \ L(A,W' \ ,\sigma)$

Stage 2
To approximate E*(C) and X*(C) we use one step gradient descent to update E and C.

$$\begin{aligned} E' &= E - \xi_E \nabla_E [L(E, X, D^{(\mathrm{tr})}_{er}) + \gamma L(E, X, \sigma(C, E, D_b))] \\ X' &= X - \xi_X \nabla_X [L(E, X, D^{(\mathrm{tr})}_{er}) + \gamma L(E, X, \sigma(C, E, D_b))] \end{aligned} \tag{18}$$

Stage 3
Plugging in these approximations to stage 3

$$L(A, W', \sigma(C, E', D_b))/|\sigma(C, E', D_b)| - \lambda L(E', X', D^{(\mathrm{val})}_{er}) \tag{19}$$

Aim: To maximise the objective in (19) using gradient based methods.

Algorithm

Algorithm 2 Optimization algorithm for learning by passing tests

```
while not converged do
    1. Update the architecture of the testee by descending the gradient calculated in Eq.(16)
    2. Update the test creator of the tester by ascending the gradient calculated in Eq.(20-24)
    3. Update the data encoder and target-task executor of the tester using Eq.(18)
    4. Update the weights of the testee by descending $\nabla_W L(A, W, D_{ee}^{(\mathrm{tr})})$
end
```

Experiments

The researchers use LPT for neural architecture search similar to Liu et al.

Datasets

The CIFAR-10 (train: 50k, test: 10k, classes:10) and CIFAR-100 (train: 50k, test: 10k, classes:100) dataset is used to perform 'architecture search'
The ImageNet dataset (train: 1.2 M, test: 50k classes: 1k) is used to perform architecture evaluation
The researchers essentially search for the target cell using CIFAR datasets, use the searched cell to create a network and train/validate on the Imagenet dataset.

Experimental setup

The researchers apply their method to different NAS methods like DARTS, P-DARTS, DARTS+ and DARTS-.

Observations

1. Applying LPT to different baselines in NAS yields better results. For example, applying the method to P-DARTS, the error reduces from 17.49% to 16.28% .
2. Previous NAS methods involved a validation set to evaluate the learner. The drawback in this approach is that the learner can achieve a good performance by performing well on the majority of the easy samples thus giving a good score. But this doesn't assure that it'll be able to perform in new test cases. Contradictory to this, Skillearn ensures wholesome learning of the learner through tests which make it robust to new scenarios.
3. It is also shown that teacher architectures that are deeper are able to 'teach' better than their less complex counterparts because complex networks can learn better representations since they have more layers.

4. The method LPT-R18-P-DARTS where the teacher architecture is Resnet-18 performs the best among all the other methods
5. The number of weights and parameters associated with the method are at par with other NAS baseline methods demonstrating that LPT is able to search more efficiently.

The authors also carry out ablation studies to evaluate the robustness of the models.

Interleaving Learning (IL)

Instantiate skillearn to formalize a human learning technique- Interleaving learning.

kth learner performs learning

$$\underbrace{l_1, l_2, \cdots, l_K}_{\text{Round } 1} \underbrace{l_1, l_2, \cdots, l_K}_{\text{Round } 2} \cdots \underbrace{l_1, l_2, \cdots, l_K}_{\text{Round } m} \cdots \underbrace{l_1, l_2, \cdots, l_K}_{\text{Round } M} \tag{27}$$

first learn l1, then l2 and so on

Encoder weights similar for consecutive learners

Elements in a learner (Sample task, Image classification)
Data encoders: ______

1. The data encoder is a CNN extracting visual features of the input images.
2. The data encoders of all learners share the same architecture, but may have different weight parameters.

Task specific head:

1. The task specific head is a multi-layer perceptron which takes the visual features of an image and predicts the class label of the particular image

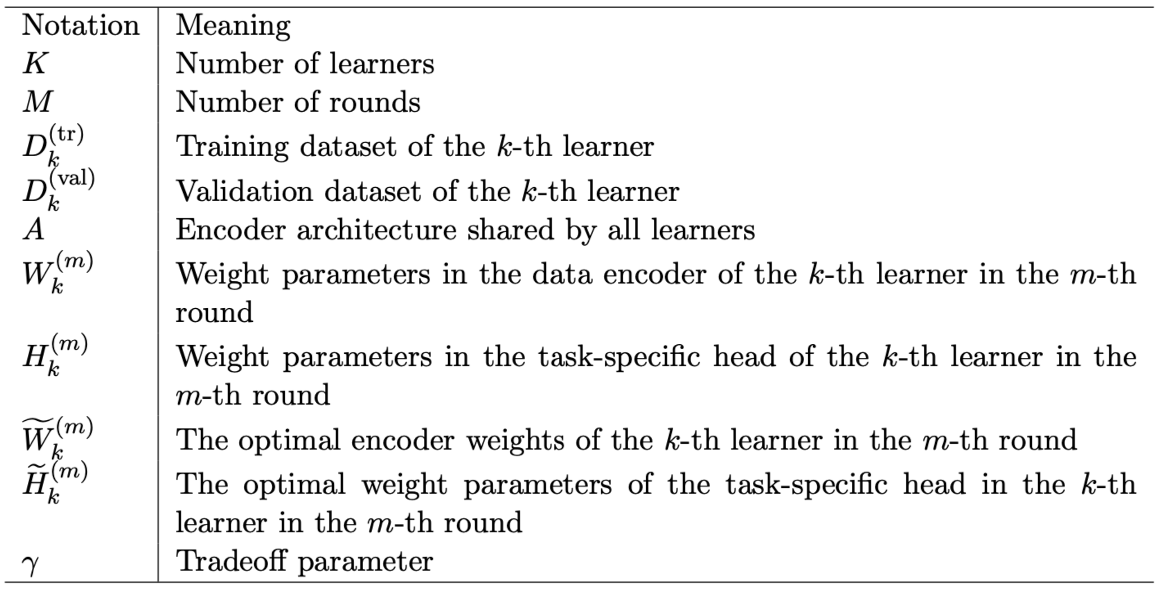
2. The architecture of the task specific head is manually decided, hence each learner has a different architecture for the task specific head and corresponding weights are also different.

In addition to this each learner has its own training and validation dataset.

Learnable parameter sets
1. architecture A of the encoder (supporting)
2. For each round m, the learner's encoder has a set of weight parameters $W_{(m)}$ specific to this round (active)
3. For each round m, the learner's task-specific head has a set of weight parameters $H_{(m)}$ specific to this round. (active)

Method

Learning stage



Notation	Meaning
K	Number of learners
M	Number of rounds
$D_k^{(\text{tr})}$	Training dataset of the k-th learner
$D_k^{(\text{val})}$	Validation dataset of the k-th learner
A	Encoder architecture shared by all learners
$W_k^{(m)}$	Weight parameters in the data encoder of the k-th learner in the m-th round
$H_k^{(m)}$	Weight parameters in the task-specific head of the k-th learner in the m-th round
$\widetilde{W}_k^{(m)}$	The optimal encoder weights of the k-th learner in the m-th round
$\widetilde{H}_k^{(m)}$	The optimal weight parameters of the task-specific head in the k-th learner in the m-th round
γ	Tradeoff parameter

Table 12: Notations in interleaving learning

Stage 1
In each of the M rounds, each of the K learners is learned in a stage.
Learning stages= M (rounds) X K (learners)
Training loss for a learner is the target task's loss defined on the training dataset.
Optimisation problem:

$$\widetilde{W}_1^{(1)}(A) = \min_{W_1^{(1)},H_1^{(1)}} L(A, W_1^{(1)}, H_1^{(1)}, D_1^{(\mathrm{tr})}) \tag{29}$$

Stage I

$$\underset{\substack{\uparrow \\ \text{Optimal weights} \\ \text{for kth learner} \\ \text{mth round}}}{\widetilde{W}_k^{(m)}} = \min_{W_k^{(m)},H_k^{(m)}} L(A, \underbrace{W_k^{(m)}, H_k^{(m)}, D_k^{(\mathrm{tr})}}_{\substack{\uparrow \\ \text{Minimize training loss}}}) + \lambda\|\underbrace{W_k^{(m)} - \widetilde{W}_{l-1}(A)}_{\substack{\uparrow \\ \text{Interaction function}}}\|_2^2 \tag{30}$$

Note:

1. The encoder architecture is not updated at this learning stage
2. In the round of 1 to M − 1, the optimal heads are discarded after learning. In the round of M, the optimal heads are retained and will be used in the validation stage.

Active learners	The k-th learner
Active learnable parameters	Weights of the data encoder and weights of the task-specific head in the k-th learner
Supporting learnable parameters	Encoder architecture shared by all learners
Active training datasets	Training dataset of the k-th learner
Active auxiliary datasets	–
Training loss	The k-th learner trains the weights of its data encoder and the weights of its task-specific head on its training dataset: $L(A, W_k^{(m)}, H_k^{(m)}, D_k^{(\mathrm{tr})})$
Interaction function	The learner encourages its encoder weights to be close to the optimal encoder weights $\widetilde{W}_{l-1}$ learned in the $l-1$ stage: $\|W_k^{(m)} - \widetilde{W}_{l-1}\|_2^2$
Optimization problem	$\widetilde{W}_k^{(m)} = \min_{W_k^{(m)},H_k^{(m)}} L(A, W_k^{(m)}, H_k^{(m)}, D_k^{(\mathrm{tr})}) + \lambda\|W_k^{(m)} - \widetilde{W}_{l-1}(A)\|_2^2$

Table 14: Learning stage l with the k-th learner at the m-th round, in interleaving learning

Validation stage

The validation loss is the sum of every learner's validation loss calculated using the optimal encoder weights and head weights learned in the final round.

Optimisation problem:

$$\min_A \sum_{k=1}^{K} L(A, \widetilde{W}_k^{(M)}(A), \widetilde{H}_k^{(M)}(A), D_k^{(\mathrm{val})}). \tag{31}$$

To summarise,

From bottom to top, the K learners perform M rounds of interleaving learning. Learners in adjacent learning stages are coupled via the interaction function. The architecture A is not updated in the learning stages. It is learned by minimizing the validation loss

$$\begin{array}{ll}
\min_A & \sum_{k=1}^{K} L(A, \widetilde{W}_k^{(M)}(A), \widetilde{H}_k^{(M)}(A), D_k^{(\mathrm{val})}) \\
s.t. & \textbf{Round M}: \\
& \widetilde{W}_K^{(M)}(A), \widetilde{H}_K^{(M)}(A) = \min_{W_K^{(M)}, H_K^{(M)}} \quad L(A, W_K^{(M)}, H_K^{(M)}, D_K^{(\mathrm{tr})}) + \lambda \| W_K^{(M)} - \widetilde{W}_{K-1}^{(M)}(A) \|_2^2 \\
& \dots \\
& \widetilde{W}_1^{(M)}(A), \widetilde{H}_1^{(M)}(A) = \min_{W_1^{(M)}, H_1^{(M)}} \quad L(A, W_1^{(M)}, H_1^{(M)}, D_1^{(\mathrm{tr})}) + \lambda \| W_1^{(M)} - \widetilde{W}_K^{(M-1)}(A) \|_2^2 \\
& \dots \\
& \textbf{Round 2}: \\
& \widetilde{W}_K^{(2)}(A) = \min_{W_K^{(2)}, H_K^{(2)}} \quad L(A, W_K^{(2)}, H_K^{(2)}, D_K^{(\mathrm{tr})}) + \lambda \| W_K^{(2)} - \widetilde{W}_{K-1}^{(2)}(A) \|_2^2 \\
& \dots \\
& \widetilde{W}_1^{(2)}(A) = \min_{W_1^{(2)}, H_1^{(2)}} \quad L(A, W_1^{(2)}, H_1^{(2)}, D_1^{(\mathrm{tr})}) + \lambda \| W_1^{(2)} - \widetilde{W}_K^{(1)}(A) \|_2^2 \\
& \textbf{Round 1}: \\
& \widetilde{W}_K^{(1)}(A) = \min_{W_K^{(1)}, H_K^{(1)}} \quad L(A, W_K^{(1)}, H_K^{(1)}, D_K^{(\mathrm{tr})}) + \lambda \| W_K^{(1)} - \widetilde{W}_{K-1}^{(1)}(A) \|_2^2 \\
& \dots \\
& \widetilde{W}_k^{(1)}(A) = \min_{W_k^{(1)}, H_k^{(1)}} \quad L(A, W_k^{(1)}, H_k^{(1)}, D_k^{(\mathrm{tr})}) + \lambda \| W_k^{(1)} - \widetilde{W}_{k-1}^{(1)}(A) \|_2^2 \\
& \dots \\
& \widetilde{W}_2^{(1)}(A) = \min_{W_2^{(1)}, H_2^{(1)}} \quad L(A, W_2^{(1)}, H_2^{(1)}, D_2^{(\mathrm{tr})}) + \lambda \| W_2^{(1)} - \widetilde{W}_1^{(1)}(A) \|_2^2 \\
& \widetilde{W}_1^{(1)}(A) = \min_{W_1^{(1)}, H_1^{(1)}} \quad L(A, W_1^{(1)}, H_1^{(1)}, D_1^{(\mathrm{tr})})
\end{array} \tag{32}$$

Optimisation Algorithm

Optimisation problem:

$$\widetilde{W}_k^{(m)}(A) = \min_{W_k^{(m)}, H_k^{(m)}} \quad L(A, W_k^{(m)}, H_k^{(m)}, D_k^{(\mathrm{tr})}) + \lambda\|W_k^{(m)} - \widetilde{W}_{k-1}^{(m)}(A)\|_2^2$$

We approximate $W_k^{(m)}(A)$ by one step gradient descent as before. For the kth learner the approximation is,

$$\widetilde{W}_k^{(m)}(A) \approx \overline{W}_k^{(m)}(A) = W_k^{(m)} - \eta\nabla_{W_k^{(m)}} L(A, W_k^{(m)}, H_k^{(m)}, D_k^{(\mathrm{tr})}) - 2\eta\lambda(W_k^{(m)} - \overline{W}_{k-1}^{(m)}(A)) \tag{35}$$

Similarly for the optimal weight parameters of the task specific head the one step gradient descent update will be,

$$\widetilde{H}_k^{(M)}(A) \approx \overline{H}_k^{(M)}(A) = H_k^{(M)}(A) - \eta\nabla_{H_k^{(M)}(A)} L(A, W_k^{(M)}, H_k^{(M)}, D_k^{(\mathrm{tr})}) \tag{36}$$

In the validation stage we plug in these approximations of W and H to update the encoder architecture A.

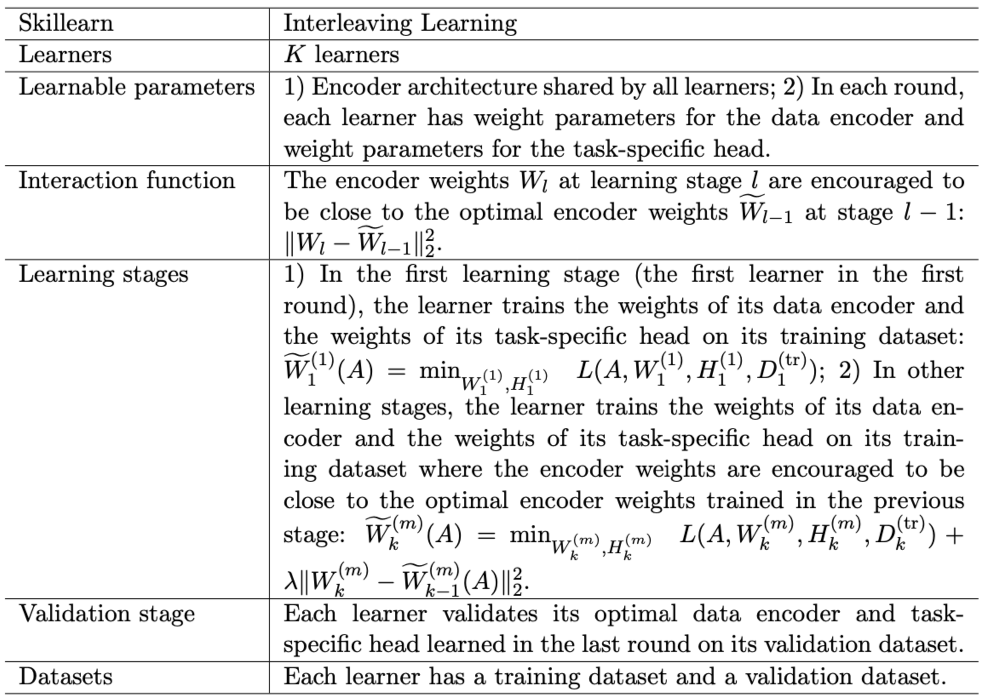
$$A \leftarrow A - \eta\sum_{k=1}^{K} \nabla_A L(A, \overline{W}_k^{(M)}(A), \overline{H}_k^{(M)}(A), D_k^{(\mathrm{val})}) \tag{37}$$

Algorithm

Algorithm 3 Optimization algorithm for interleaving learning

```
while not converged do
    1. Update $\widetilde{W}_1^{(1)}(A)$ using Eq.(34)
    2. For $k = 2 \cdots K$, update $\widetilde{W}_k^{(1)}(A)$ using Eq.(35)
    3. For $k = 1 \cdots K$ and $m = 2 \cdots M$, update $\widetilde{W}_k^{(m)}(A)$ using Eq.(35)
    4. For $k = 1 \cdots K$, update $\widetilde{H}_k^{(M)}(A)$ using Eq.(36)
    5. Update $A$ using Eq.(37)
end
```

1. Update the encoder weights of the learners at the first stage of first round
2. Update the encoder weights of the K learners in the first round
3. Update encoder weights of all the K learners for M rounds
4. Update the weights of the task specific head for the k learners
5. Update the architecture of the encoder in the validation stage.



Skillearn	Interleaving Learning
Learners	K learners
Learnable parameters	1) Encoder architecture shared by all learners; 2) In each round, each learner has weight parameters for the data encoder and weight parameters for the task-specific head.
Interaction function	The encoder weights W_l at learning stage l are encouraged to be close to the optimal encoder weights $\widetilde{W}_{l-1}$ at stage $l-1$: $\|W_l - \widetilde{W}_{l-1}\|_2^2$.
Learning stages	1) In the first learning stage (the first learner in the first round), the learner trains the weights of its data encoder and the weights of its task-specific head on its training dataset: $\widetilde{W}_1^{(1)}(A) = \min_{W_1^{(1)}, H_1^{(1)}} \; L(A, W_1^{(1)}, H_1^{(1)}, D_1^{(\mathrm{tr})})$; 2) In other learning stages, the learner trains the weights of its data encoder and the weights of its task-specific head on its training dataset where the encoder weights are encouraged to be close to the optimal encoder weights trained in the previous stage: $\widetilde{W}_k^{(m)}(A) = \min_{W_k^{(m)}, H_k^{(m)}} \; L(A, W_k^{(m)}, H_k^{(m)}, D_k^{(\mathrm{tr})}) + \lambda\|W_k^{(m)} - \widetilde{W}_{k-1}^{(m)}(A)\|_2^2$.
Validation stage	Each learner validates its optimal data encoder and task-specific head learned in the last round on its validation dataset.
Datasets	Each learner has a training dataset and a validation dataset.

Table 16: Mapping from Skillearn to Interleaving Learning

Experiments
The authors apply interleaving learning on the task of NAS.

Datasets
Two tasks are performed, image classification on CIFAR-100 and image classification on CIFAR-10 using two classifiers A and B.
CIFAR-10: 25k training set ($D_A^{(tr)}$) and 25k ($D_A^{(val)}$) validation set
CIFAR-100: 25k training set ($D_B^{(tr)}$) and 25k ($D_B^{(val)}$) validation set
Imagenet (as described in LPT)

Experimental setup
The method is in the lines of Liu et al consisting of a search and evaluation phase.
The authors experiment with the search spaces of DARTS, P-DARTS, PC-DARTS. During architecture search, two rounds of learning are performed, with an order of CIFAR- 100, CIFAR-10, CIFAR-100, CIFAR-10. During architecture evaluation, copies of the searched cell are used to create a network and in turn trained and evaluated using the Imagenet dataset.

Results

1. Applying interleaving learning brings a significant reduction in error rates against baseline approaches.
2. Using interleaving learning, the search task of CIFAR-10 and CIFAR-100 mutually benefit each other as encoder weights trained on the first task are used to initialise the encoder weights for the second task, which is better than random initialisation of weights.
3. The researchers also demonstrate that IL obtains better results as compared to Joint learning approaches.
4. Finally IL is efficient as it does not increase the number of parameters or search cost for the task.

Key insights and enabling ideas

1. The authors develop skillearn, a machine learning framework inspired by human learning skills (HLS) and leverage it to train better ML models.
2. The framework consists of multiple learner models (having learnable parameters like weights, etc) that interact with each other using interaction functions
3. The learning process is divided into stages each involving a subset of learners. Stages are performed end to end in a multilevel optimization framework.
4. The stages can influence each other i.e the latter stages affect the earlier stages and vice versa,
5. They have used the generalised framework to mimic two HLS namely, learning by passing tests and interleaving learning
6. In LPT, a tester model dynamically creates tests to evaluate a testee model.
7. The tester model consists of three elements, the data encoder, the test creator and the target task executor that work together to create better tests in order to make the testee/learner more robust The testee model learns to solve more difficult tests.
8. Experiments show that the LPT framework gives promising results for the task of neural architecture search.
9. In interleaving learning, the learners switch between learning tasks that improve retention and the ability to transfer learned knowledge.
10. A set of models collaboratively learn a data encoder in an interleaving fashion. The encoder is trained by model A for a while, then passed to model B for further training, then model C, and so on. This process repeats for some specified number of rounds.
11. Experiments of neural architecture search on CIFAR-100, CIFAR-10 and Imagenet datasets demonstrate the effectiveness of interleaving learning.

Strengths

1. The model is successfully able to capture the abstract concept of human learning skills i.e the techniques humans employ to improve retention.
2. It seems that Skillearn is able to provide comparable error rates to those reported by its predecessors on the CIFAR-10/100 dataset with significant reduction in the number of parameters.
3. The skillearn framework is flexible as it can be applied to any differentiable NAS method.
4. Most of the previous approaches aimed at improving differential NAS by updating the network architecture (network pruning) but skillearn differs in its approach as it mimics HLS to improve the learning process of machine learning models.

Potential areas of improvement

1. Can we take the student models capacity into account, whether it is capable of learning to such an extent?