## Motivation

Hyperparameter optimization is important but often done haphazardly.
- **Insufficient exploration** may lead to poor results, killing a promising research idea
- **Inequitable allocation** of compute for hyperparameter optimization may lead to exaggerated differences among models

We need tools to standardize the process and make things easy for researchers.

**Contribution:** a toolkit for optimizing Neural Machine Translation transformer models (in Sockeye3 framework) on a distributed grid

https://github.com/kevinduh/sockeye-recipes3

**Example:** High variance in model accuracy & speed due to different hyperparameters. The tool finds good models automatically.

![700+ Transformer models on a sw-en Machine Translation task](image)

**Hyperparameter optimization is important but often done haphazardly.**
- **Insufficient exploration** may lead to poor results, killing a promising research idea
- **Inequitable allocation** of compute for hyperparameter optimization may lead to exaggerated differences among models

We need tools to standardize the process and make things easy for researchers.

**Contribution:** a toolkit for optimizing Neural Machine Translation transformer models (in Sockeye3 framework) on a distributed grid

https://github.com/kevinduh/sockeye-recipes3

**Example:** High variance in model accuracy & speed due to different hyperparameters. The tool finds good models automatically.

![700+ Transformer models on a sw-en Machine Translation task](image)

## Problem Formulation

### Hyperparameter Optimization (HPO):

Given a fixed budget of “function evaluations”, find as many Pareto-optimal hyperparameter settings \( x \) as possible

**Hyperparameter setting encoded as vector in \( \mathbb{R}^d \):**

\[
\begin{align*}
3 & \rightarrow \# \text{ layers} \\
200 & \rightarrow \# \text{ units/layer} \\
1 & \rightarrow \text{ optimizer type} \\
0.2 & \rightarrow \text{ learning rate}
\end{align*}
\]

**Accuracy:** \( f_1(x) \)

**Inference Speed:** \( f_2(x) \)

**Train Model on dataset**

### Definition:

Assume we want to find \( x \) that maximizes \( f_1(x) \) and \( f_2(x) \). A point \( p \) is **pareto-optimal** iff there does not exist a \( q \) such that \( f_k(q) \geq f_k(p) \) for all \( k \) and \( f_k(q) > f_k(p) \) for at least one \( k \).

### Software Design

1. **User defines hyperparameter space**
2. Sample a subset of configurations. These are candidates for training on the compute grid.
3. Run hyperparameter optimization, which intelligently decides whether or when to train each config given budget

## Specific Implementation: ASHA

**Many hyperparameter optimization methods:**
- Bayesian Optimization
- Evolutionary Algorithms
- Population-based Training
- Bandit Learning

**See our EACL23 tutorial:**

We implement a bandit method called **ASHA** (Asynchronous Successive Halving Algo):
- Trains multiple config in parallel
- After few checkpoints, pre-emptively stop training for models that under-perform
- **Assume:** learning curves are comparable
- **Resources are spent on promising config**