Are Transformers Good Learners?

Exploring the Limits of Transformer Training

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Neural Networks vs Humans

- Current SotA networks can achieve “human-like” performance.
  - language modeling (GPT-3), machine translation (CUBBITT)

SYSTEM PROMPT (HUMAN-WRITTEN)	***In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.***
MODEL COMPLETION	*The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These **four-horned, silver-white unicorns** were previously unknown to science.* *Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.* *[...]*

Table 12 from “Language Models are Unsupervised Multitask Learners”, Radford et al. (2018)

Neural Networks vs Humans

- Current SotA networks can achieve “human-like” performance.
  - language modeling (GPT-3), machine translation (CUBBITT)
- Inspired by biological neurons, but far from biological learning:
  - very narrowly specialized (vs general knowledge)
  - require huge amount of training examples
  - cannot exploit previous knowledge efficiently

Outline

1. Generalization in NLP
2. Catastrophic Forgetting
3. Modularity and Knowledge Composition

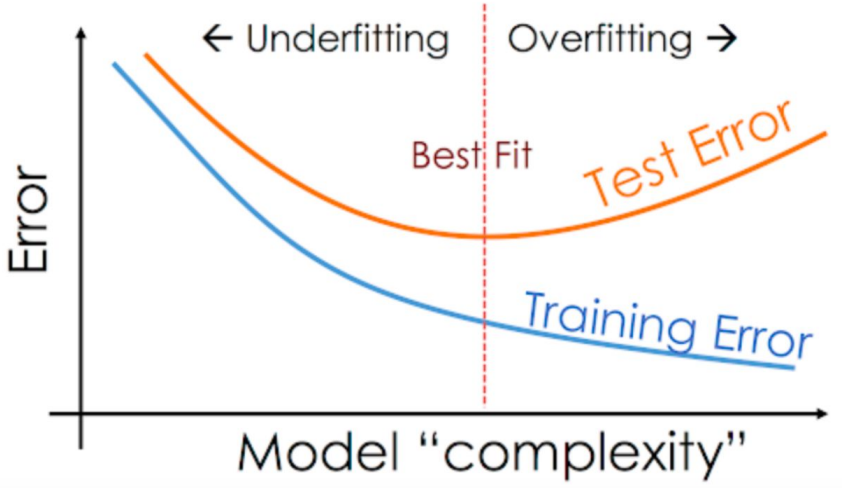
Generalization in NLP

Generalization in Machine Learning

- We want to measure the ability to “solve” new instances of a problem.
  - in practice, split data into **train**, **development** and **test** datasets
  - the splits should have **similar distribution** (e.g. similar text domain)

Generalization in Machine Learning

- We want to measure the ability to “solve” new instances of a problem.
  - in practice, split data into **train**, **development** and **test** datasets
  - the splits should have **similar distribution** (e.g. similar text domain)
- We measure generalization to avoid overfitting model to **train** dataset.
  - we do not want a model that just “memorizes” training data
  - occam’s razor principle



Generalization and NLP

- How do we define **novel instances** of a problem?
  - Ideally, we want to guarantee zero overlap with the training data.
  - Just use examples not present in our training data.
  - What about partial overlap between training and validation instances?

Generalization and NLP

- How do we define **novel instances** of a problem?
  - Ideally, we want to guarantee zero overlap with the training data.
  - Just use examples not present in our training data.
  - What about partial overlap between training and validation instances?
- What if there are multiple solutions to an instance of a problem?
  - one solution in train and the other in test dataset
  - E.g. (Machine Translation): I saw a man with a telescope.
    - => “**Viděl** jsem muže **skrze dalekohled**.” (I saw a man through a telescope.)
    - => “**Viděl** jsem muže **s dalekohledem.**” (I saw a man “carrying” a telescope.)

Generalization and NLP

- Partial overlap can be a problem in NLP:
  - may lead to overestimation of the model performance
  - Train: “A **cat** sat on a mat.“
  - Test: “A **dog** sat on a mat.”

Generalization and NLP

- Partial overlap can be a problem in NLP:
  - may lead to overestimation of the model performance
  - Train: “A **cat** sat on a mat.“
  - Test: “A **dog** sat on a mat.”
- Possible solutions:
  - Multiple random splits (Gorman and Bedrick, 2019)
  - Adversarial splits (Søgaard et al., 2020)

Generalization and NLP

- GPT-3: SotA language modeling
  - billions of training tokens, multiple domains
  - also hundreds of billions model weights

Generalization and NLP

- GPT-3: SotA language modeling
  - billions of training tokens, multiple domains
  - also hundreds of billions model weights
- we are still learning its capabilities and limits
  - (no) internal memory
    - (AI Dungeon - Text based RPG) [play.aidungeon.io]
    - GPT-3: “You have entered a cave. There is a **sleeping dragon** protecting a treasure.”
    - Player: “I loot the treasure under the **dragon’s corpse.”**
    - GPT-3: “You found a sword under the **dragon’s corpse**.”

Generalization and NLP

- GPT-3: SotA language modeling
  - billions of training tokens, multiple domains
  - also hundreds of billions model weights
- we are still learning its capabilities and limits
  - (no) internal memory
    - (AI Dungeon - Text based RPG) [play.aidungeon.io]
    - GPT-3: “You have entered a cave. There is a **sleeping dragon** protecting a treasure.”
    - Player: “I loot the treasure under the **dragon’s corpse.”**
    - GPT-3: “You found a sword under the **dragon’s corpse**.”
  - arithmetics (Brown et al., 2020)
    - two-digit number addition/multiplication
    - it might be just memorizing addition tables

Generalization and NLP

- GPT-3: SotA language modeling
  - billions of training tokens, multiple domains
  - also hundreds of billions model weights
- we are still learning its capabilities and limits
  - (no) internal memory
    - (AI Dungeon - Text based RPG) [play.aidungeon.io]
    - GPT-3: “You have entered a cave. There is a **sleeping dragon** protecting a treasure.”
    - Player: “I loot the treasure under the **dragon’s corpse.”**
    - GPT-3: “You found a sword under the **dragon’s corpse**.”
  - arithmetics (Brown et al., 2020)
    - two-digit number addition/multiplication
    - it might be just memorizing addition tables
  - reasoning (Digital Philosopher) [www.alphai.cz/digitalni-filosof/]
    - we might be simply asking questions that are not uncommon in philosophy literature.
    - e.g. : “What is your opinion on killing animals?”

Toy Tasks And Model Analysis

- Removing train/test overlap could lead to much clearer model analysis.

Toy Tasks And Model Analysis

- Removing train/test overlap could lead to much clearer model analysis.
- String Editing:
  - “Edit op” + separator + sequence of “a” and “b”
  - Edit op: copy, push X, pop, unshift X, shift, reverse, flip
  - only one correct answer for each *operation+string* combination
  - these task are easy to generalize for humans

Toy Tasks And Model Analysis

- Removing train/test overlap could lead to much clearer model analysis.
- String Editing:
  - “Edit op” + separator + sequence of “a” and “b”
  - Edit op: copy, push X, pop, unshift X, shift, reverse, flip
  - only one correct answer for each *operation+string* combination
  - these task are easy to generalize for humans
- Examples:
  - “push a | a b b a” => “a b b a a”
  - “reverse - | b a b a” => “a b a b”
  - “flip - | a b b b a” => “b a a a b”

Toy Tasks And Model Analysis

- Removing train/test overlap could lead to much clearer model analysis.
- String Editing:
  - “Edit op” + separator + sequence of “a” and “b”
  - Edit op: copy, push X, pop, unshift X, shift, reverse, flip
  - only one correct answer for each *operation+string* combination
  - these task are easy to generalize for humans
- Examples:
  - “push a | a b b a” => “a b b a a”
  - “reverse - | b a b a” => “a b a b”
  - “flip - | a b b b a” => “b a a a b”
- Hypothesis:
  - Transformer can learn to solve new instances of a task regardless of the length of the instance.

String Editing: Experiment Setup

1. Generate set of unique (a, b) sequences

String Editing: Experiment Setup

1. Generate set of unique (a, b) sequences
2. Split to buckets based on input length:
    - **10** (0-10), **15** (11-15), **20** (16-20)

String Editing: Experiment Setup

1. Generate set of unique (a, b) sequences
2. Split to buckets based on input length:
   - **10** (0-10), **15** (11-15), **20** (16-20)
3. Split each bucket to Train, Devel and Test sets
   - use only **15-bucket for training (30k examples)**

String Editing: Experiment Setup

1. Generate set of unique (a, b) sequences
2. Split to buckets based on input length:
   - **10** (0-10), **15** (11-15), **20** (16-20)
3. Split each bucket to Train, Devel and Test sets
   - use only **15-bucket for training (30k examples)**
4. Add a task label given the examined task
   - Separate evaluation for each task...
   - … vs joint training on all tasks (using 15-bucket Train only)

String Editing: Experiment Setup

- Network:
  - encoder-decoder Transformer (Fairseq “transformer” setting)
  - depth 1
  - 1 attention head
  - embedding size 128
  - 100 training epochs
- Evaluation metric == Accuracy (exact string match)

String Editing: Experiment Setup

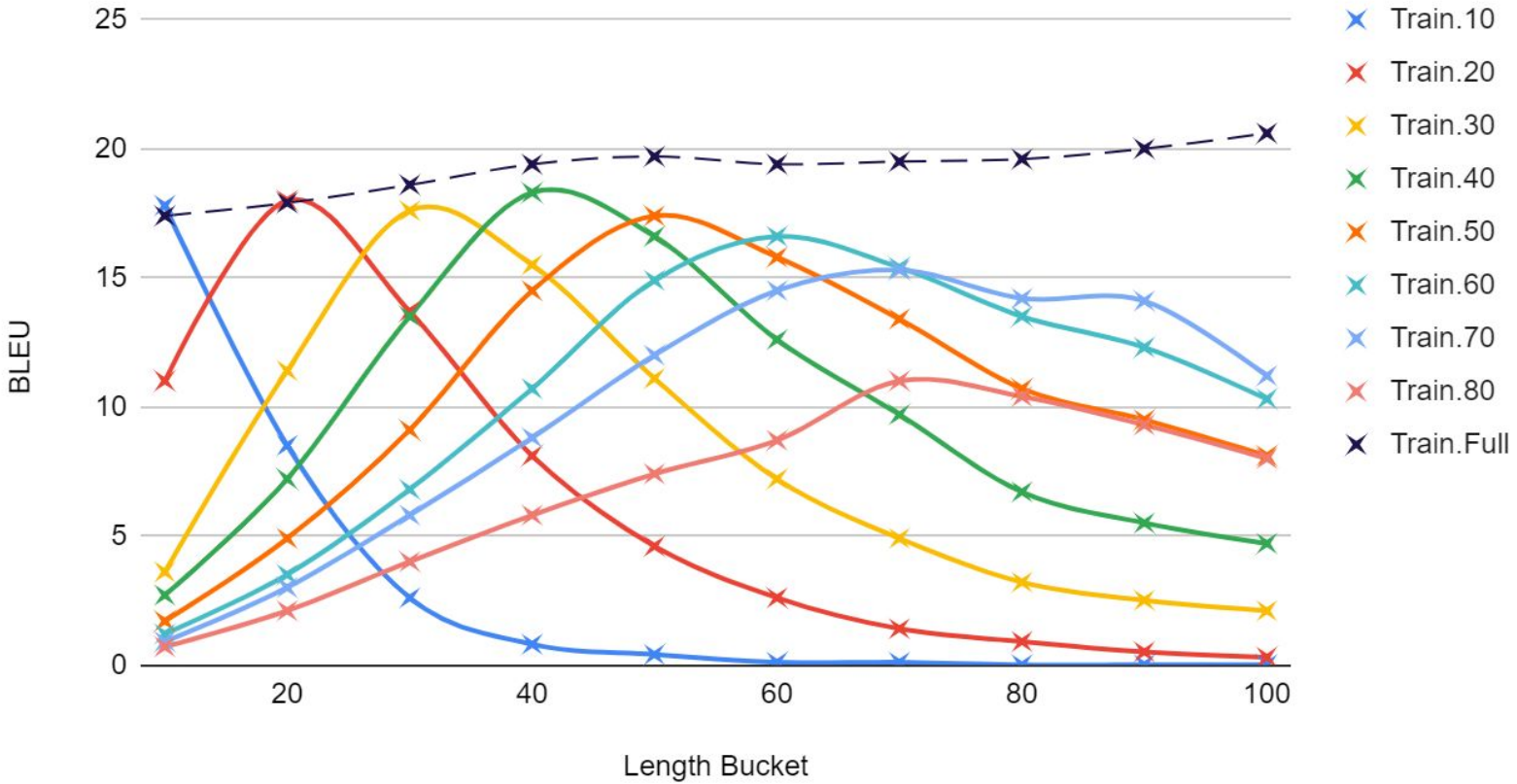
- Network:
  - encoder-decoder Transformer (Fairseq “transformer” setting)
  - depth 1
  - 1 attention head
  - embedding size 128
  - 100 training epochs
- Evaluation metric == Accuracy (exact string match)

ACC (%)	copy	push	pop	shift	unshift	reverse	all
10-bucket	59,1	72,5	0,0	62,4	55,9	0,9	38,4
15-bucket	**100,0**	**100,0**	**100,0**	**100,0**	**100,0**	**62,6**	**96,0**
20-bucket	0,0	0,0	0,0	0,0	0,0	0,0	2,0

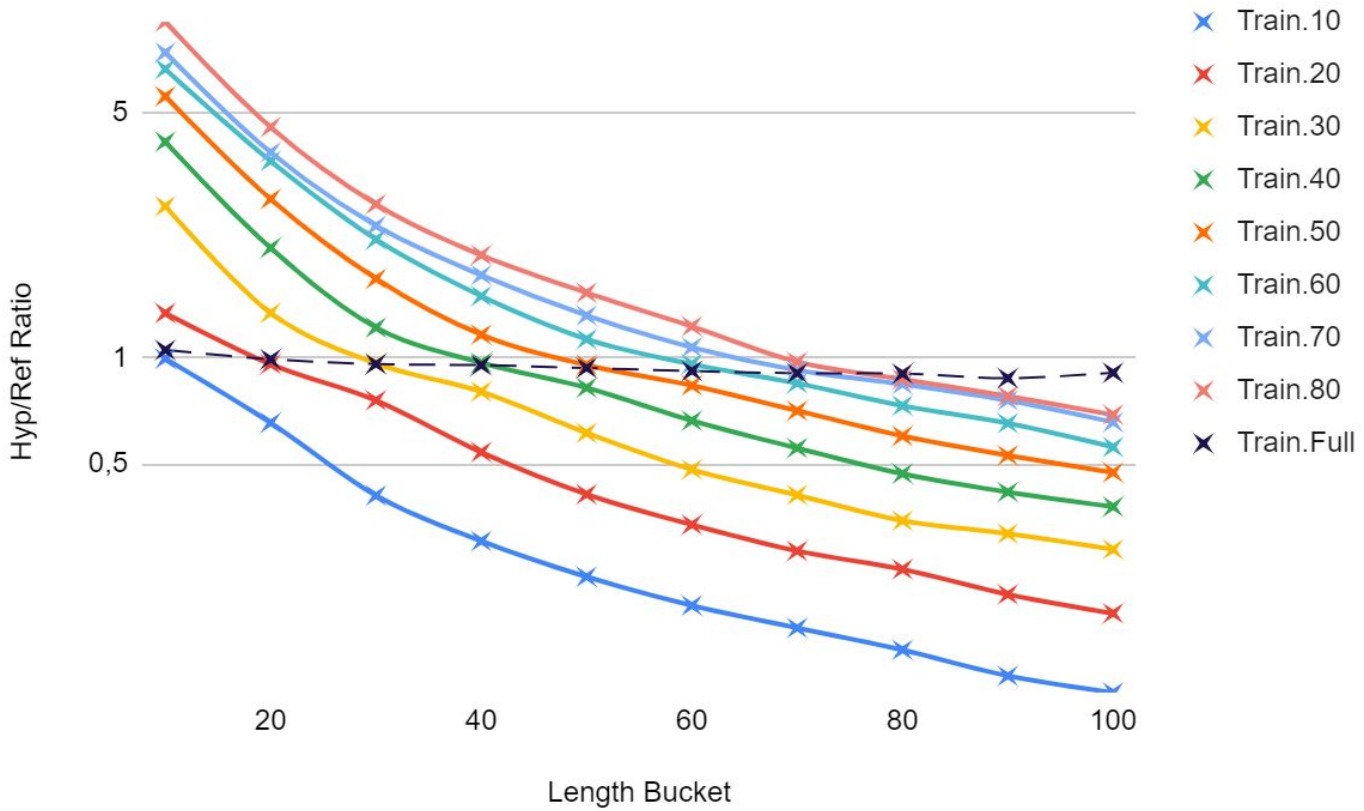
Length Generalization: Machine Translation

- Similar setup to String Edit:
  - 6 layer Transformer, 8 attention heads, 512 embedding size
- Training - CzEng 2.0 (en => cs):
  - Moses tokenization, 30k BPE
  - **Split to buckets based on TGT (cs) length (10-bucket, …, 100-bucket)**
- Validation:
  - WMT newstest13-16 (no buckets) + early stopping (BLEU)
- Test:
  - WMT newstest17-20 (bucketed)

Length Generalization: Machine Translation



Length Generalization: Machine Translation



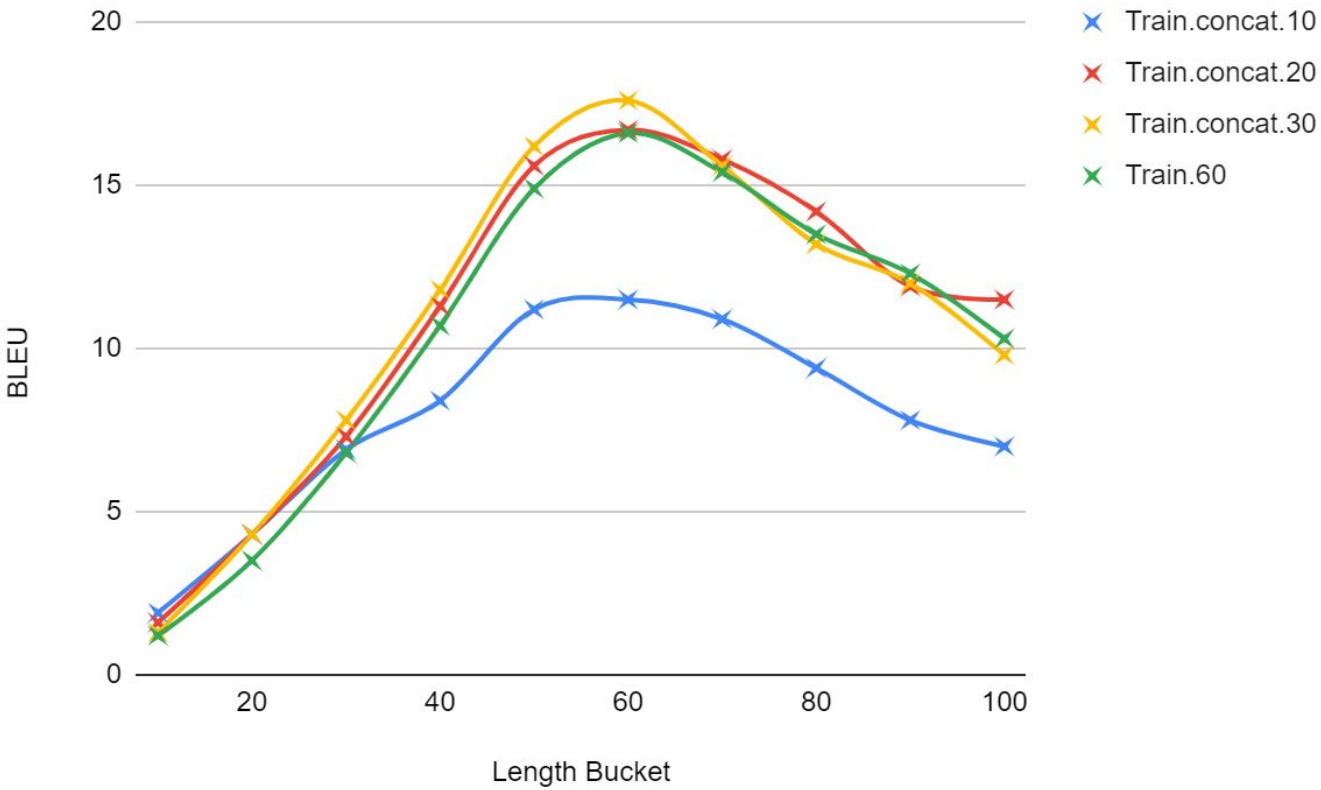
Length Overfitting

- Overfitting to training length is a problem…
  - … but can we take advantage of it?

Length Overfitting

- Overfitting to training length is a problem…
  - … but can we take advantage of it?
- Data augmentation by concatenation
  - We take 10-bucket (or 20-bucket, 30-bucket) and concatenate training examples to create datasets of tgt length 51-60
  - Simple sliding window method
  - Comparison to genuine 60-bucket

Length Overfitting: Data Augmentation



Catastrophic Forgetting

Catastrophic Forgetting

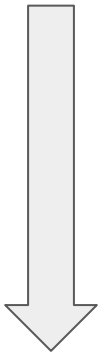
- A good AI system should be able to update its knowledge continuously
  - humans also start with “knowledge priors” when learning new tasks
  - NLP: learning new languages, adaptation to new domains
  - Incremental Learning
- Deep networks suffer from Catastrophic forgetting
  - (McCloskey and Cohen, 1989)
  - Optimal weights of the original task are disregarded when learning new tasks

Forgetting and String Edit Task

- Train the Transformer in a sequential manner
  - Copy => Push => Pop => Shift => Unshift => Reverse
  - 15-bucket
  - each task - 100 epochs
  - compare to joint training
  - single encoder-decoder layer, 8 attention heads

Forgetting: String Editing

- Previous tasks completely “forgotten” when trained without any constraints



train \ test	copy	push	pop	shift	unshift	reverse
copy	100	0	0	0	0	0,8
push	0	100	0	0	0	0
pop	0	0	100	0,1	0	0
shift	0	0	0,1	100	0	0
unshift	0	0	0	0	100	0
reverse	0,8	0	0	0	0	84,4
all	100	100	100	100	100	97,5

Avoiding Catastrophic Forgetting

- Avoid significant updates to task-important weights
- Regularization-based methods
  - restrict updates to weights that are crucial for the previous tasks
  - Elastic Weight Consolidation (EWC, Kirkpatrick et al., 2017)
  - Synaptic Intelligence (Zenke et al., 2017)

Avoiding Catastrophic Forgetting

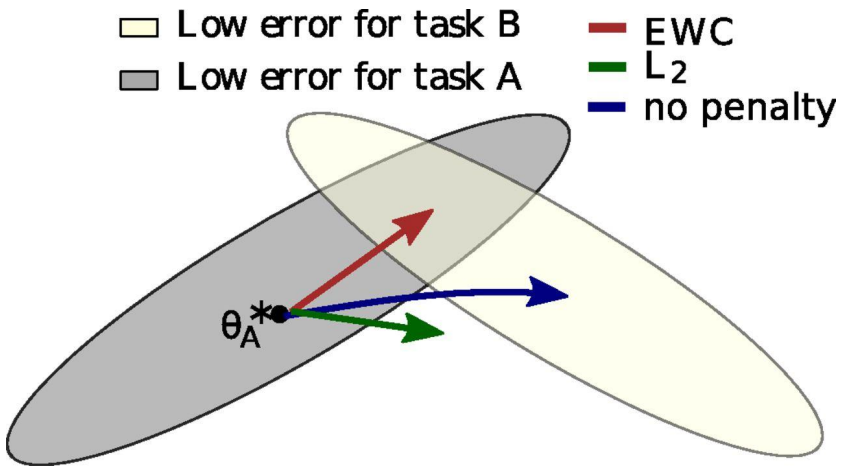
- Avoid significant updates to task-important weights
- Regularization-based methods
  - restrict updates to weights that are crucial for the previous tasks
  - Elastic Weight Consolidation (EWC, Kirkpatrick et al., 2017)
  - Synaptic Intelligence (Zenke et al., 2017)
- Adapter-based methods
  - freeze the original weights + add a trainable subnetwork
  - BERT (Devlin et al., 2017), MT adaptation via adapters (Bapna and Firat, 2019)

Avoiding Catastrophic Forgetting

- Avoid significant updates to task-important weights
- Regularization-based methods
  - restrict updates to weights that are crucial for the previous tasks
  - Elastic Weight Consolidation (EWC, Kirkpatrick et al., 2017)
  - Synaptic Intelligence (Zenke et al., 2017)
- Adapter-based methods
  - freeze the original weights + add a trainable subnetwork
  - BERT (Devlin et al., 2017), MT adaptation via adapters (Bapna and Firat, 2019)
- Replay-based methods
  - periodically expose model to samples from previously learned tasks
  - generative replay (van de Ven et al., 2020)

Elastic Weight Consolidation (EWC)

- Unsupervised Pre-training Using Elastic Weight Consolidation (Kirkpatrick et al., 2017)
- Experiments on permuted MNIST
  - new task = single input pixel permutation applied to the whole MNIST dataset
  - comparison to no regularization and L2 regularization



EWC: Motivation

Datasets for task A and B:

$$\mathcal{D} = \mathcal{D}_{\mathcal{A}} \cup \mathcal{D}_{\mathcal{B}}$$

Joint learning:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

EWC: Motivation

Datasets for task A and B:

$$\mathcal{D} = \mathcal{D}_{\mathcal{A}} \cup \mathcal{D}_{\mathcal{B}}$$

Joint learning:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

Learning task B after task A:

$$p(\theta|\mathcal{D}_{\mathcal{B}}) = \frac{p(\mathcal{D}_{\mathcal{B}}|\theta)p(\theta|\mathcal{D}_{\mathcal{A}})}{p(\mathcal{D}_{\mathcal{B}})}$$

EWC: Motivation

Datasets for task A and B:

$$\mathcal{D} = \mathcal{D}_{\mathcal{A}} \cup \mathcal{D}_{\mathcal{B}}$$

Joint learning:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

Learning task B after task A:

$$p(\theta|\mathcal{D}_{\mathcal{B}}) = \frac{p(\mathcal{D}_{\mathcal{B}}|\theta)p(\theta|\mathcal{D}_{\mathcal{A}})}{p(\mathcal{D}_{\mathcal{B}})}$$

How do we compute $p(\theta|D_A)$?

EWC: Regularization Derivation

- Computing $p(\theta|D_A)$ is intractable.
- We estimate it as a multivariate Gaussian:
  - based on the work on Laplace approximation (MacKay, 1992)
  - Mean: parameter values at the end of task A
  - Variance (diagonal only): diagonal of Fisher Information matrix

EWC: Regularization Derivation

- Computing $p(\theta|D_A)$ is intractable.
- We estimate it as a multivariate Gaussian:
  - based on the work on Laplace approximation (MacKay, 1992)
  - Mean: parameter values at the end of task A
  - Variance (diagonal only): diagonal of Fisher Information matrix
- Fisher Information matrix:
  - equivalent to 2nd derivation near local extrema
  - can be computed from first derivations
  - Positive-semidefinite
  - in practice, we use Empirical Fisher Information (approx. of FI)

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathcal{B}}(\theta) + \sum_i \frac{\lambda}{2} F_i (\theta_i - \theta_{\mathcal{A}}, i)$$

EWC: Unsupervised Pretraining

- Can we use EWC regularization to pretrain parts of networks?
- Unsupervised Pretraining for Neural Machine Translation Using Elastic Weight Consolidation (Variš and Bojar, 2019):
  - low-resource MT (small amount of bilingual data, larger monolingual corpora)
  - avoid overfitting to bi-data by “remembering” language modeling on source-side and target-side

Unsupervised Pretraining: Setup

- Can we use EWC regularization to pretrain parts of networks?
- Unsupervised Pretraining for Neural Machine Translation Using Elastic Weight Consolidation (Variš and Bojar, 2019):
  - low-resource MT (small amount of bilingual data, larger monolingual corpora)
  - avoid overfitting to bi-data by “remembering” language modeling on source-side and target-side
- Data
  - Basque-to-English (Eu-En) Translation
  - Bilingual: IWSLT18 TED Talks (9,1k sent-pairs) + General domain data (940k sent-pairs)
  - Monolingual: Basque Wikipedia (3M sentences), WMT News Commentary (3M sentences)
- Model:
  - Transformer (NeuralMonkey)
  - 6 layers, 512 embeddings size, 16 attention heads

Pretraining

- Transformer decoder LM (empty encoder output) for both Eu and En
  - left-to-right decoding
  - encoder-decoder attention removed
  - minimizing cross-entropy (train and dev) + early-stopping
- FI estimation:
  - on IWSLT development dataset (respective parts)
  - process development data in batches and compute mean gradient

Experiments:

- 3 contrastive systems:
  - baseline transformer (no pretraining)
  - train + backtranslated monolingual data
  - LM objective regularization (Ramachandran et al., 2017)
- LM regularization:
  - MT loss combined with pretraining loss
  - same pretraining as ours

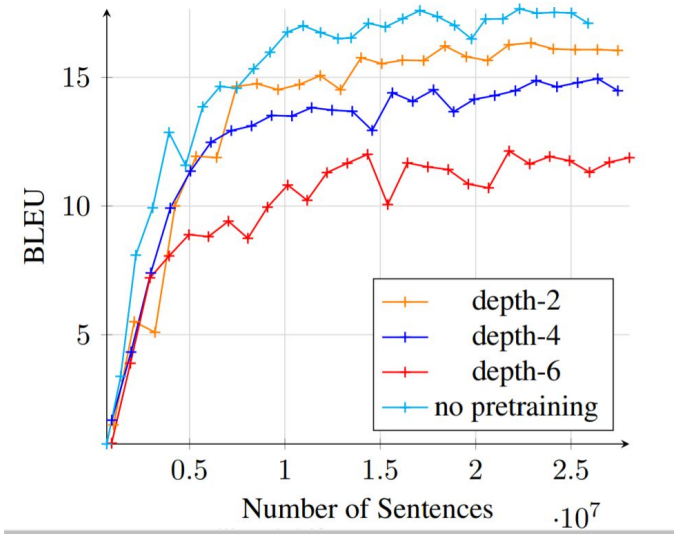
Results

- Backtranslation wins
- Pretraining and regularizing decoder-only seems promising

BLEU (x100)		SRC	TGT	ALL
Baseline	15.68	--	--	--
Backtranslation	**19.65**	--	--	--
LM reg.	--	13.96	15.56	16.83
EWC reg.	--	10.77	**15.91**	14.10
LM reg. (ens)	--	15.16	16.60	17.14
EWC reg. (ens)	--	10.73	**16.63**	14.66

Further Analysis: Encoder Pretraining

- Possible incompatibility between LM decoder and MT encoder
  - LM: only leftward context
  - MT: left-and-right context



Modularity and Knowledge Composition

Knowledge Composition

- Creating complex concepts from a simple ones
  - Humans can create infinite number of sentences from a fixed set of rules (Chomsky, 1965)

Knowledge Composition

- Creating complex concepts from a simple ones
  - Humans can create infinite number of sentences from a fixed set of rules (Chomsky, 1965)
- Current networks - Using whole network
  - “A taste” of composition - multi-head attention, mixture-of-experts
  - Still requires execution of the whole network

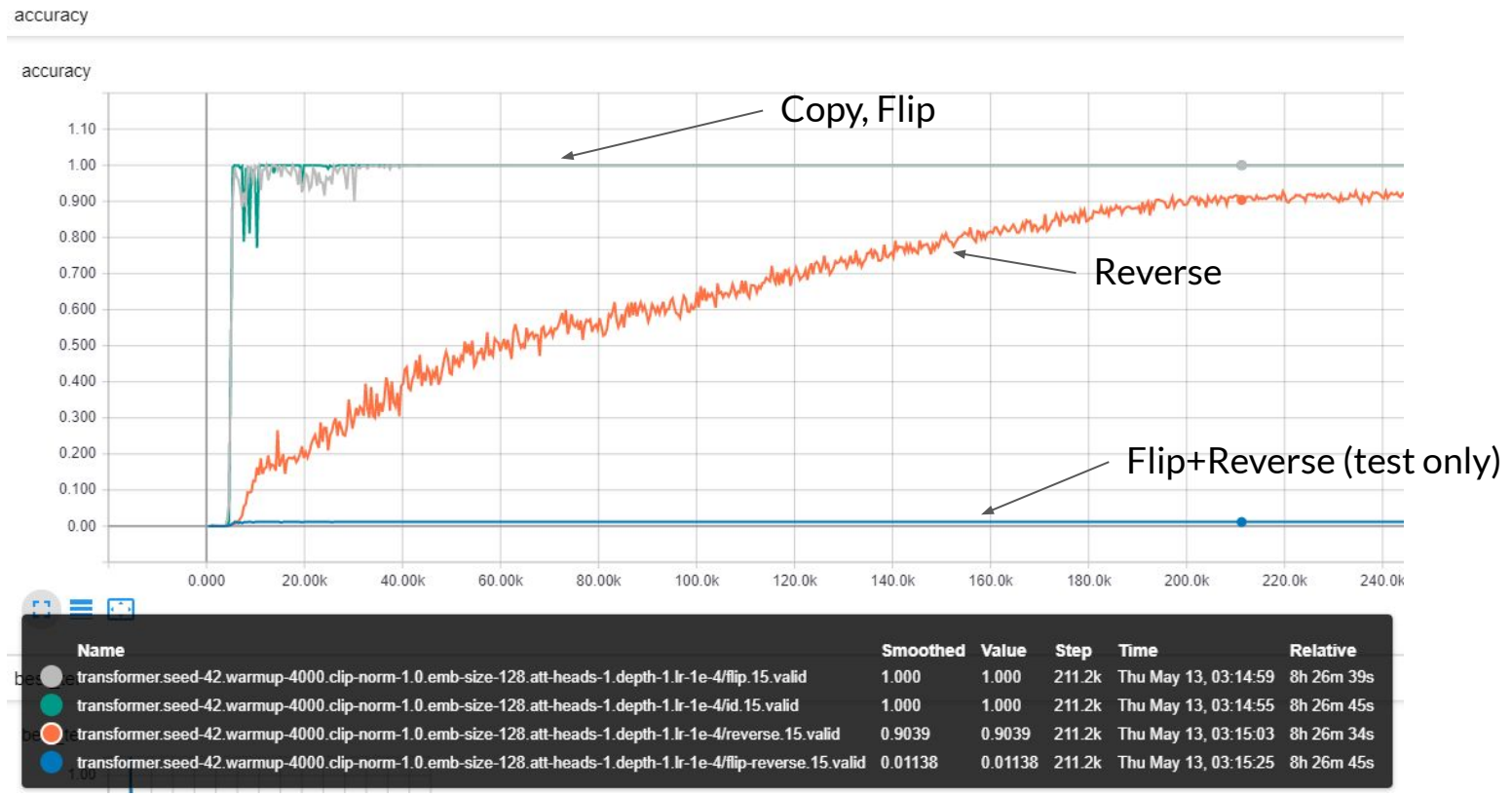
Knowledge Composition

- Creating complex concepts from a simple ones
  - Humans can create infinite number of sentences from a fixed set of rules (Chomsky, 1965)
- Current networks - Using whole network
  - “A taste” of composition - multi-head attention, mixture-of-experts
  - Still requires execution of the whole network
- Adapter-based can also be considered a knowledge composition
  - However, the user has to switch the adapters based on the task at hand
  - Ideally, the network itself would choose which subnetworks need to be executed

Composition: String Editing

- Instead of using labels (copy, reverse, etc.) we encode them in binary flags
  - Two tasks (flip and reverse)
  - “0” = off, “1” = on
- Basic configurations:
  - Flip (“a” to “b”, “b” to “a”) = “1 0” (training, test)
  - Reverse = “0 1” (training, test)
- “Derived” configurations:
  - Copy = “0 0” (training, test)
  - Flip+Reverse = “1 1” (test only)
- Examples:
  - “0 1 | b a b a” => “a b a b”
  - “1 0 | a b b b a” => “b a a a b”

Composition: String Editing



Name	Smoothed	Value	Step	Time	Relative
transformer.seed-42.warmup-4000.clip-norm-1.0.emb-size-128.att-heads-1.depth-1.lr-1e-4/flip.15.valid	1.000	1.000	211.2k	Thu May 13, 03:14:59	8h 26m 39s
transformer.seed-42.warmup-4000.clip-norm-1.0.emb-size-128.att-heads-1.depth-1.lr-1e-4/id.15.valid	1.000	1.000	211.2k	Thu May 13, 03:14:55	8h 26m 45s
transformer.seed-42.warmup-4000.clip-norm-1.0.emb-size-128.att-heads-1.depth-1.lr-1e-4/reverse.15.valid	0.9039	0.9039	211.2k	Thu May 13, 03:15:03	8h 26m 34s
transformer.seed-42.warmup-4000.clip-norm-1.0.emb-size-128.att-heads-1.depth-1.lr-1e-4/flip-reverse.15.valid	0.01138	0.01138	211.2k	Thu May 13, 03:15:25	8h 26m 45s

Network Modularization

- The whole network has to be executed regardless of the task
  - task distinction (probably) only via hidden representations
  - insufficient for composition of multiple tasks

Network Modularization

- The whole network has to be executed regardless of the task
  - task distinction (probably) only via hidden representations
  - insufficient for composition of multiple tasks
- Conditional Execution of Subnetworks
  - based on Modular networks (Kirsch et al., 2018)
  - our case: modularization of multi-head attention

Attention Modularization

- After training some attention heads can be pruned:
  - reducing model size without any significant drop
  - the pruning is done with respect to the whole task
  - (Michel et al., 2019, Voita et al. 2019)

Attention Modularization

- After training some attention heads can be pruned:
  - reducing model size without any significant drop
  - the pruning is done with respect to the whole task
  - (Michel et al., 2019, Voita et al. 2019)
- Pruned heads ~ wasted model capacity
  - Can we “reuse” these heads when learning multiple tasks?
  - Can we get a more interpretable attention?

Multi-head Attention

Basic formula:

$$MHA(q, x) = \sum_{i=0}^{H-1} Att_i(q, x)$$

Multi-head Attention

Basic formula:

$$MHA(q, x) = \sum_{i=0}^{H-1} Att_i(q, x)$$

i-th attention head:

$$Att_i(q, \mathbf{x}) = W_o^i \sum_{j=0}^{N-1} Softmax(\frac{q^\intercal (W_q^i)^\intercal W_k^i x_j}{\sqrt{d}}) W_v^i x_j$$

Multi-head Attention

Basic formula:

$$MHA(q, x) = \sum_{i=0}^{H-1} Att_i(q, x)$$

i-th attention head:

$$Att_i(q, \mathbf{x}) = W_o^i \sum_{j=0}^{N-1} Softmax(\frac{q^\intercal (W_q^i)^\intercal W_k^i x_j}{\sqrt{d}}) W_v^i x_j$$

Masked Multi-head Attention:

$$MHA(q, x) = \sum_{i=0}^{H-1} \xi_i Att_i(q, x)$$

(Mask variable ξ_i can have 1 or 0 values.

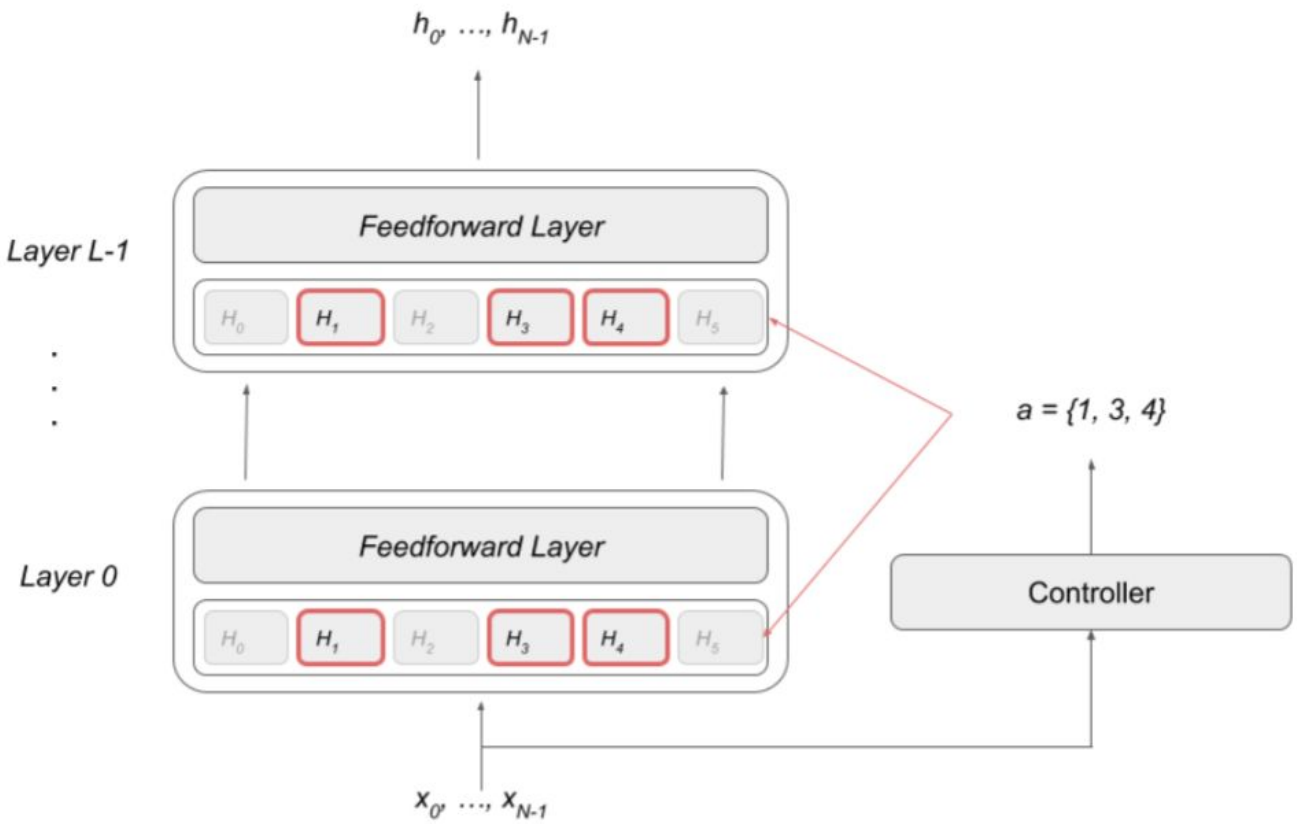
Modular Multi-head Attention

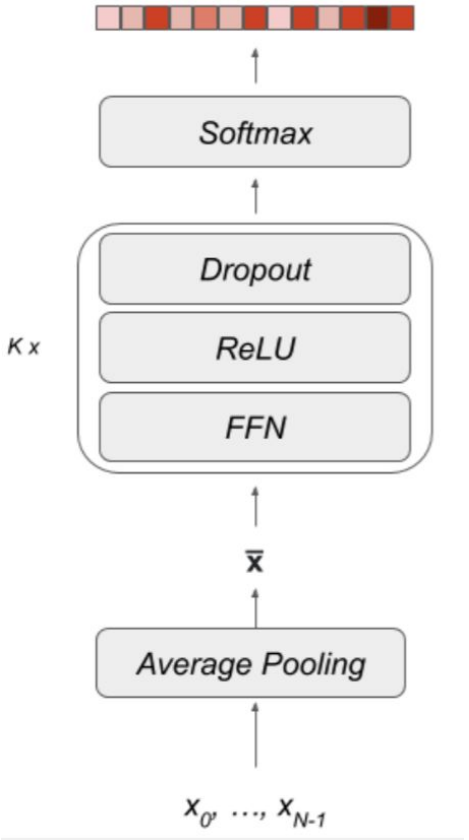
- Mask variable ξ_i can be either fixed or predicted by the network
  - "Controller" predicts the mask values based on the input
  - Our approach:
    - mask prediction based on the input sequence (embedded words)
    - same mask applied at every attention layer (encoder and decoder)

$$p(y|x,\theta,\phi) = \sum_a p(y,a|x,\theta,\phi)$$

$$p(y,a|x,\theta,\phi) = p(y|a,x,\theta)p(a|x,\phi)$$

Modular Multi-head Attention





(Controller)

Modular Multi-head Attention: Training

- Sum over all possible module combinations is intractable:
  - use Expectation-Maximization (Neal and Hinton, 1998)
  - Expectation step:
    - sample module combinations
    - compute loss for each sample
    - pick the sample with lowest loss

Modular Multi-head Attention: Training

- Sum over all possible module combinations is intractable:
  - use Expectation-Maximization (Neal and Hinton, 1998)
  - Expectation step:
    - sample module combinations
    - compute loss for each sample
    - pick the sample with lowest loss
  - Maximisation step:
    - use the best sample (so far) from expectation step for model update

Modular Multi-head Attention: Training

- Sum over all possible module combinations is intractable:
  - use Expectation-Maximization (Neal and Hinton, 1998)
  - Expectation step:
    - sample module combinations
    - compute loss for each sample
    - pick the sample with lowest loss
  - Maximisation step:
    - use the best sample (so far) from expectation step for model update
  - We do not perform Expectation step in every batch
    - we keep track of best sample for each example so far

MMHA: Experiments

- IWSLT English-to-German translation:
  - Train: IWSLT14 train data
  - Devel: IWSLT17/tst2014 (SacreBLEU)
  - Test: IWSLT17/tst2015 (SacreBLEU)
- Transformer (Fairseq)
  - embedding size 512
  - 8 attention heads
  - 4k warmup steps (no modularization during warmup)
  - early-stopping (BLEU on Devel data)

Results

iwslt17/tst2014	base	$E\text{-}freq = 1$			$E\text{-}freq = 5$		
		ENC	DEC	ALL	ENC	DEC	ALL
base	21.3	–	–	–	–	–	–
1-head	–	20.6	**21.5**	20.9	20.4	*17.6*	19.3
2-head	–	20.9	**21.7**	20.8	20.8	*21.3	19.5
4-head	–	21.1	**21.8**	***21.4**	21.1	***21.4**	20.5
iwslt17/tst2015							
base	22.5	–	–	–	–	–	–
1-head	–	21.6	***22.6**	22.0	21.5	*18.5*	20.1
2-head	–	22.0	***22.7**	21.8	21.7	*22.4	20.6
4-head	–	*22.4	**22.8**	*22.4	22.2	*22.4	21.7

Table 1: Mean translation quality (sample size 5) measured using BLEU (multiplied by 100 for better readability) of the Transformer baseline (*base*) and systems with with varying sizes of attention head subsets. Two training settings are compared based on the frequency of the expectation step: either in each mini-batch or after every five mini-batches. Numbers in **bold** show an improvement over the baseline system. Numbers in *italics* indicate a system where at least one of the sample training diverged, lowering the mean BLEU. Asterisk (*) indicates results where the difference from the respective baseline is not statistically significant ($p\text{-}value > 0.05$).

Results: Oracle Experiments

iwslt17/tst2014	w/ctrl	fixed	oracle
1-head	20.9	*20.9	*20.9
4-head	21.4	*17.4*	22.1
iwslt17/tst2015			
1-head	22.0	*21.8	*22.0
4-head	22.4	*18.3*	**23.2**

Table 2: Mean BLEU comparison (multiplied by 100, with 5 randomly initialized runs each) between the standard ModMHA Transformer (*w/ctrl*; $E\text{-}freq = 1$), a version with a single fixed subset (*fixed*) and the oracle setting (*oracle*). Each system was trained using the ALL setting. Scores in **bold** indicate an improvement over the respective w/ctrl baseline. *Italics* indicate settings, where at least one of the runs did not converge. Asterisk (*) indicates results where the difference from the respective w/ctrl baseline is not statistically significant ($p\text{-}value > 0.05$).

MMHA: Module Collapse

- Is the head-choice during training balanced?
  - some heads are picked more often during training => get updated more frequently
  - frequently updated heads can be more compatible with the previous/following layers
  - this could lead to further preference of these heads during M-step

MMHA: Module Collapse

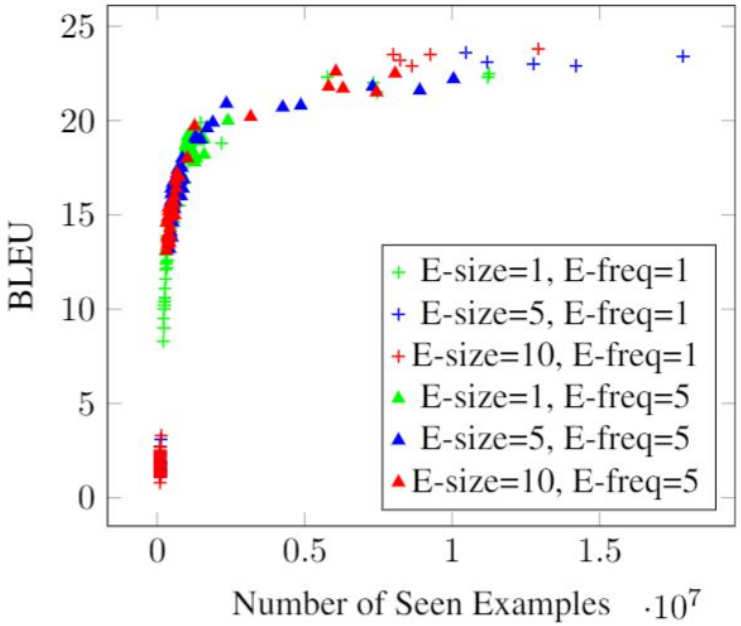


Figure 4: A correlation between the number of examples a specific head has seen during training (1-head, ALL setting) and the BLEU score of a system using only that head. Systems were evaluated on `iwslt17/tst14` dataset.

Learning in Transformer: Final Remarks

- We need to be careful about overfitting...
  - … and how to detect it during evaluation
  - sequence length distribution in training data matters
- Improving learning in Transformer is still far from solved
  - there are promising techniques available (EWC, modularization),
  - their application is not straightforward
- Knowledge composition is still problematic
  - straighforward modularization or designing special architectures?
  - some task may not even be possible to decompose to simple problems