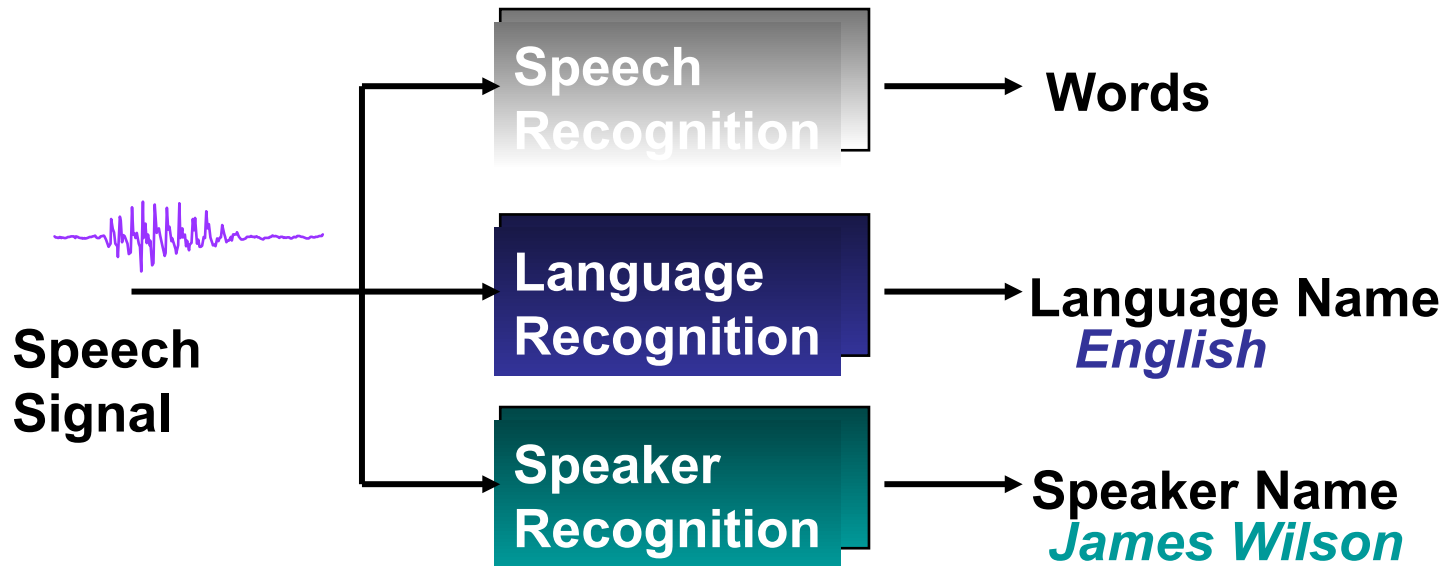


Natural Language and Speech Processing

Lecture 13: Speaker and Language Recognition.

Some General Tips & Tricks.

Extracting information from speech



Speaker recognition applications

- **Access control**
 - Physical facilities
 - Data and data networks
- **Transaction authentication**
 - Telephone credit card purchases
 - Bank wire transfers
 - Fraud detection
- **Monitoring**
 - Remote time and attendance logging
 - Home parole verification
- **Information retrieval**
 - Customer information for call centers
 - Audio indexing (lectures, podcasts, radio)
 - Personalization
- **Forensics**
 - Voice sample matching

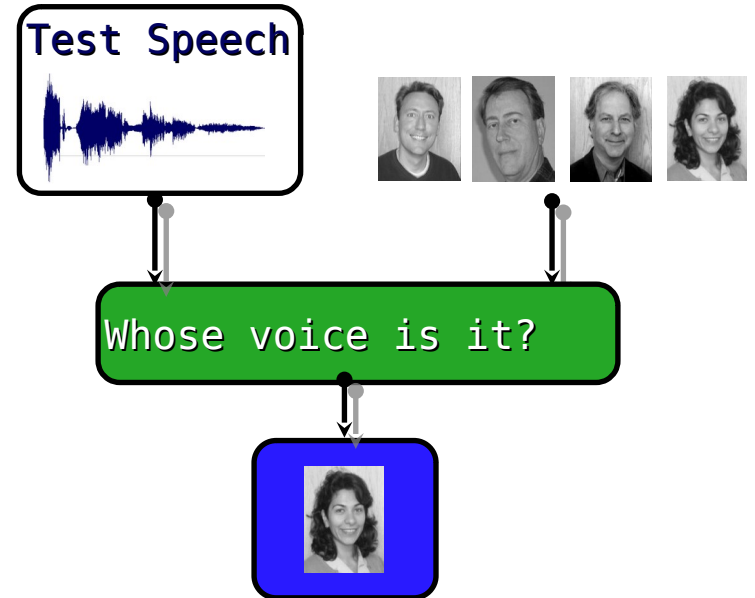


Tasks

- Speaker identification
 - Closed set and open set
- Speaker verification
- Also:
 - Speaker change detection
 - Speaker diarization (clustering of speaker segments according to speaker)
 - Speaker overlap detection

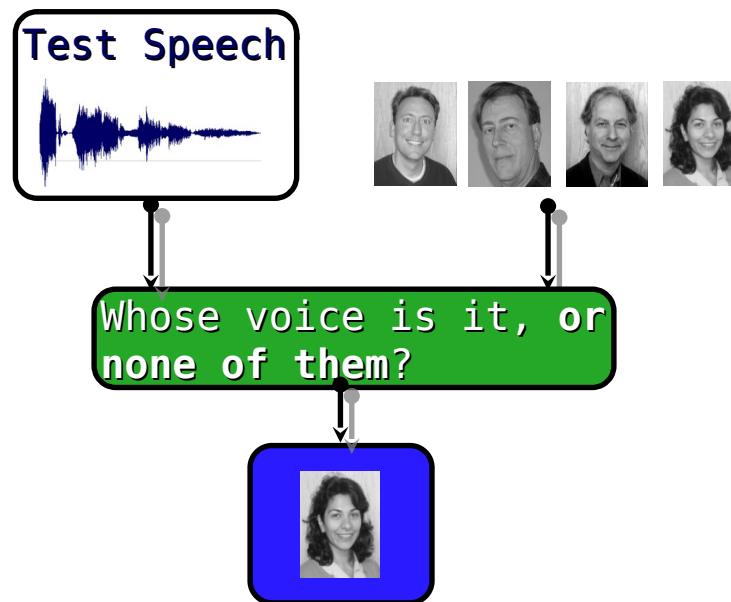
Speaker identification

- Closed set speaker identification
 - We have training data for all speakers we are interested in
 - During test time, the speaker is guaranteed to be one of those speakers



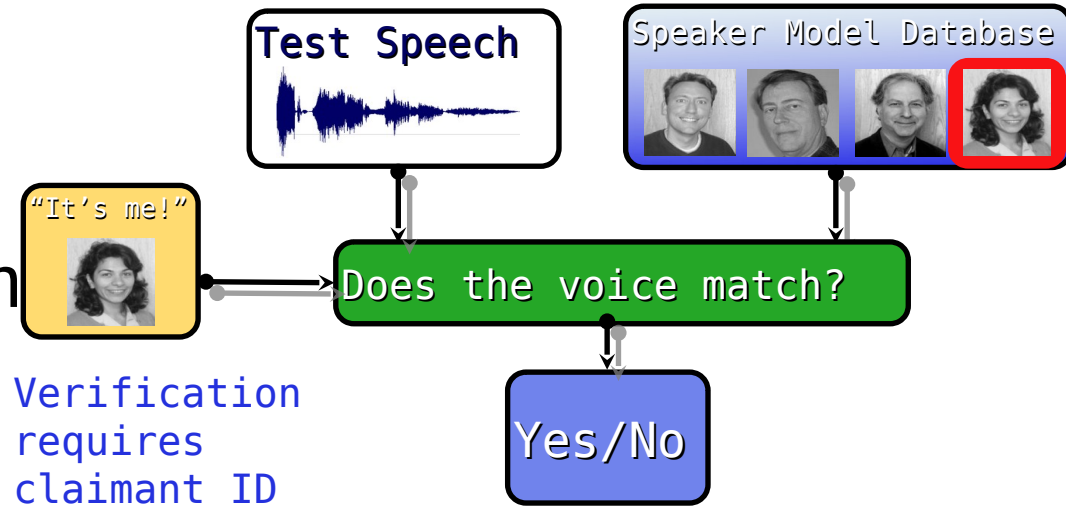
Speaker identification

- Open set speaker identification
 - We have training data for all speakers we are interested in
 - During test time, the speaker is **not** guaranteed to be one of those speakers



Speaker verification

- User claims an identity
- System task: Accept or reject identity claim
- The voice can come from outside the set of known speakers

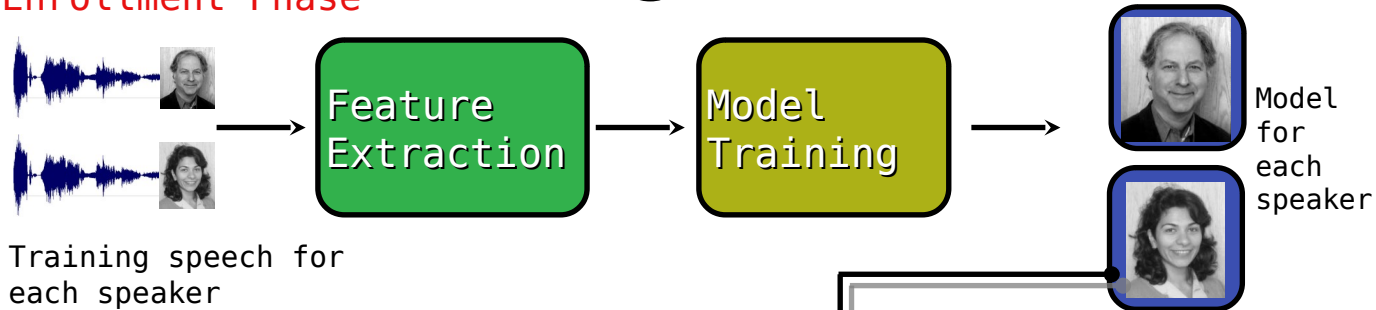


Speaker verification types

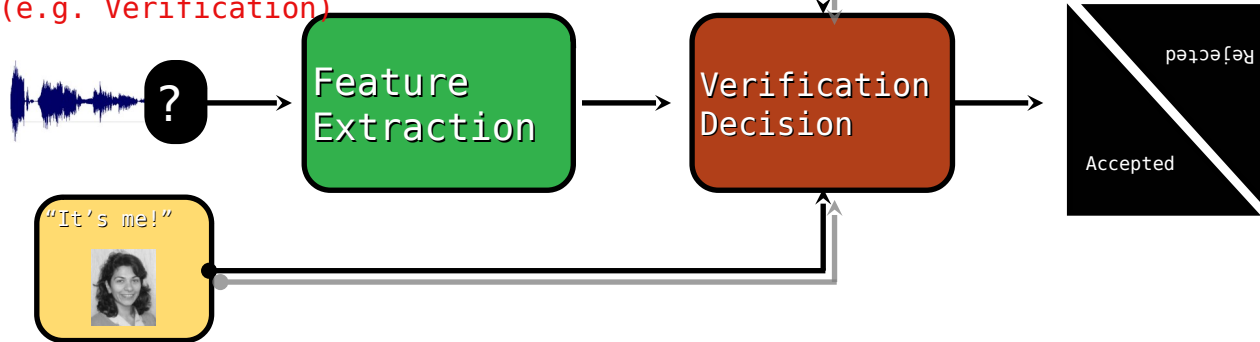
- Text-independent recognition
 - Recognition system does not know text spoken by person
 - Examples: User selected phrase, conversational speech
 - Used for applications with less control over user input
 - More flexible system but also more difficult problem
 - Speech recognition can provide knowledge of spoken text
 - For example: telephone banking – use as an additional verification measure and apply during the conversation
- Text-dependent recognition
 - Recognition system knows text spoken by person
 - Examples: password phrase, prompted phrase
 - Used for applications with strong control over user input
 - Knowledge of spoken text can improve system performance
 - Prompting may reduce risk of imposters using voice recordings (spoofing)

Verification: enrollment and recognition

Enrollment Phase

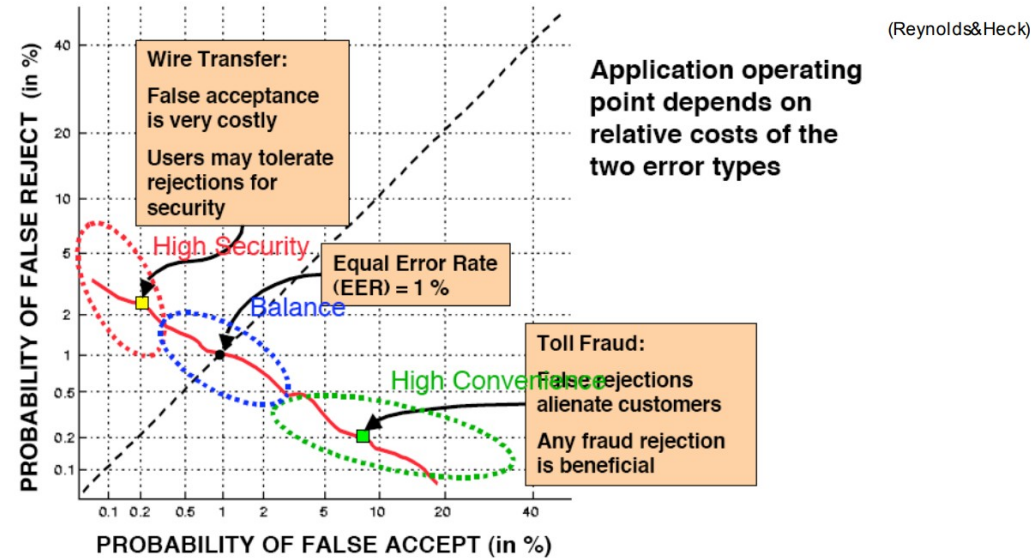


Recognition Phase (e.g. Verification)



Speaker verification operation point

- Speaker verification performance is often measured in terms of equal error rate (EER)
 - a decision threshold that produces equal number of false rejects and false accepts
- The actual decision threshold (operation point) is highly dependent on the application
 - Are false rejects or false accepts more costly?

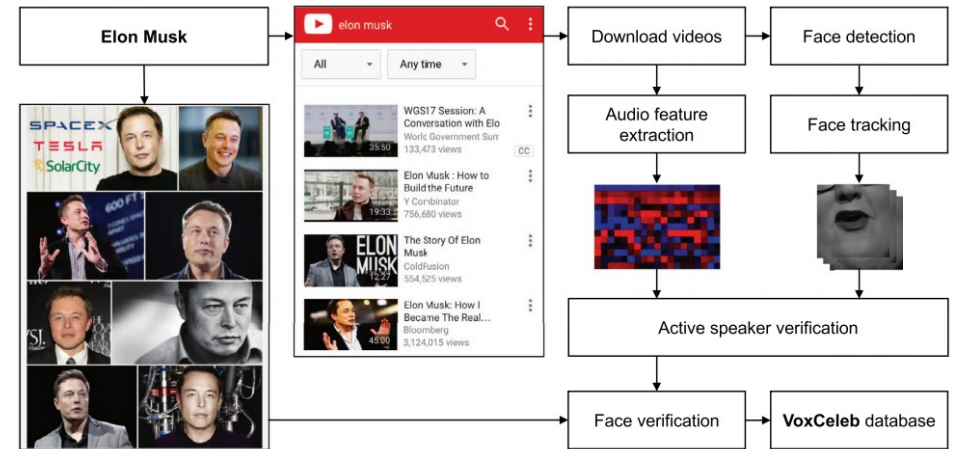


Speaker verification: modern approach

- Modern speaker recognition is purely based on pretrained models
 - Large (possibly out-of-domain) speaker-labeled speech dataset is used to train an utterance-to-speaker classifier
 - The final classifier layer is then thrown away, and the second-to-last layer output is used as an utterance embedding extractor
 - Utterance embeddings of enrolment and test utterances can now be compared
 - If they are similar (above some tuned threshold), the speaker is verified
- Research questions:
 - Where to get this large speech dataset?
 - How to reduce the effect of domain mismatch (e.g.: the embeddings are trained on English, but we want to apply it for Estonian: will it work?)
 - What is the optimal architecture of the speaker classification/embedding model and how to train it so that it will be most useful as an embedding extractor?
 - How to compute similarity between embeddings, possibly in mismatched conditions (e.g.: enrolment is silent environment, test utterance in a car)

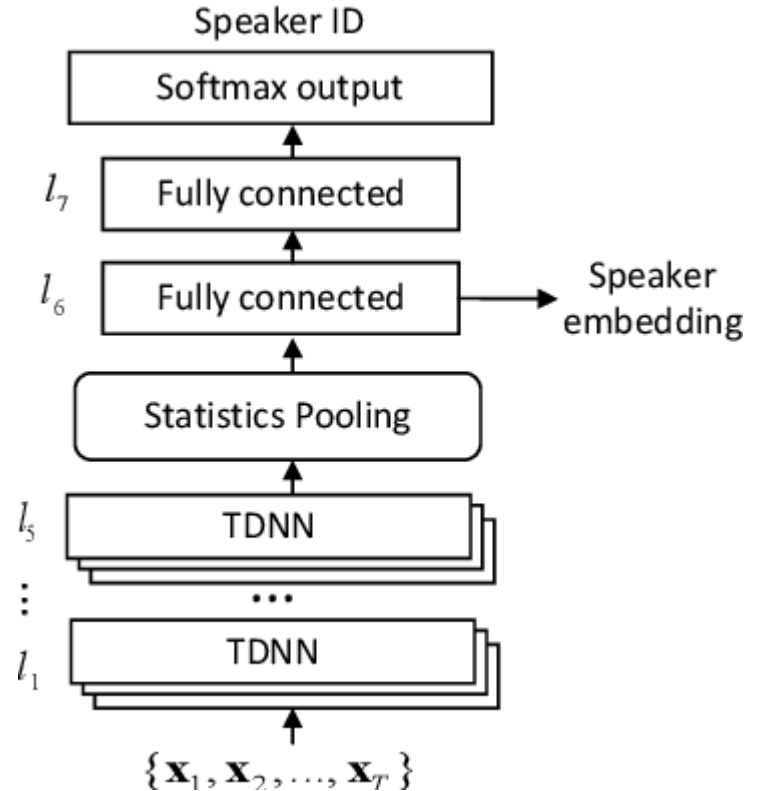
VoxCeleb dataset

- How to get a free, diverse, and very large dataset for pretraining?
- VoxCeleb (1, 2): automatically collected dataset of 7000+ speakers, 2500+ hours
- Collection process:
 - Compile a list of celebrities with known faces (based on VGGFace dataset): from actors and sportspeople to politicians
 - Search for „<name> interview” on YouTube
 - Download videos
 - Find segments in the videos where the face of <name> appears
 - Active speaker verification: determine the audio-video synchronisation between mouth motion and speech in a video
 - Extract all such segments and label them with <name>



X-vector model

- A breakthrough in speaker recognition (2018)
- Architecture:
 - Input: filterbank features (same as in speech recognition)
 - Several layers of frame level 1D convolutional layers (called TDNNs in the paper)
 - Statistics pooling layer: computes mean and stddev over the utterance of each feature extracted by the last convolutional layer (i.e., like conventional average pooling, but also adds stddev for each feature)
 - Two utterance level fully connected layers
 - Softmax
- Speaker embeddings are extracted from penultimate fully connected layer
- Must be trained with heavy data augmentation:
 - Take training data, and make several copies of it, with:
 - Take a random 3-second cut from the training utterances (makes the training more challenging for the model)
 - Random room reverberation
 - Added background noise
 - Added background music
 - Added „babble” noise
- Only works when training dataset is large enough (e.g., VoxCeleb)

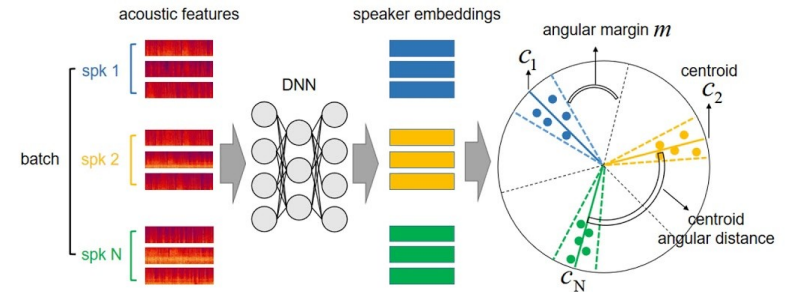


Improvements to x-vector model

- Statistics pooling can be replaced with attention
 - This way, pooling is done only using the most relevant parts of the utterance
- 1D CNNs can be replaced with 2D CNNs (essentially the same as work well in image recognition)
- Don't optimise cross-entropy loss, but the angular margin loss
 - I.e., the angle between the centroid of the speakers' utterances in the embedding space
 - Benefit: speaker verification can be done by just comparing the angles between the enrollment centroid, and the test utterance

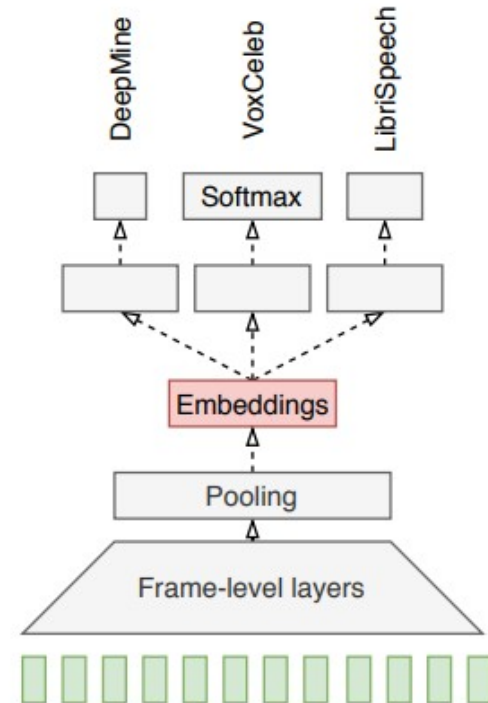
Method

Angular Margin Centroid Loss



Language portability of X-vectors

- Turns out that embeddings trained on VoxCeleb work pretty good for other languages as well
- But it's better if you add some language-specific data
- However, it's better to separate the language-specific speakers (actually, all datasets with different characteristics) during training
 - I.e., each training dataset has its own output layer, and the speakers in the training data „compete” only with other speakers of the same dataset (Alumäe & Valk, 2020)
 - This way, language/dataset specific knowledge is pushed to the dataset-specific output layers and the embeddings are „encouraged” to be universal

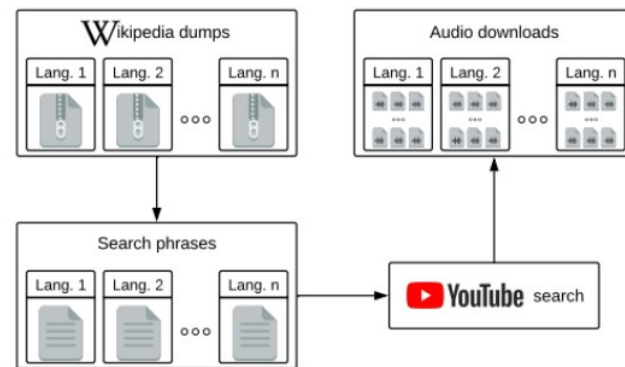


Spoken language identification

- Very similar to speaker recognition
 - X-vectors with modifications
 - Data augmentation during training
- Pretraining still helps:
 - E.g., use a very large language-labelled speech corpus to train a model for extracting embeddings
 - Only ~200 utterances per language are needed now to train a a new model for a new set of languages
 - But how/where to get this large language-labelled speech dataset?

VoxLingua107

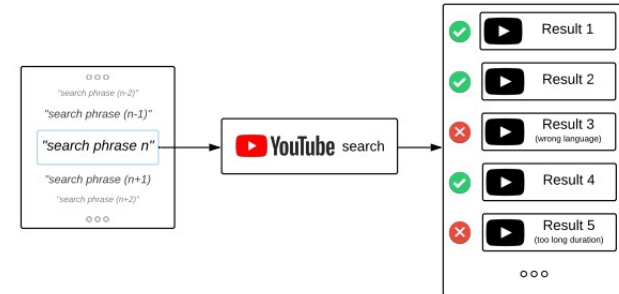
- Research done in our lab (Valk & Alumäe, 2020)
- Like VoxCeleb, but for language recognition
- A dataset of speech „from the wild” (YouTube) for 107 languages
- Collected automatically: random trigram search queries, based on Wikipedia of the particular language
- Idea: if the title and description of the video is in a particular language, then it likely contains speech from this language
- Text-based language identification model was used to filter out phrases that were not (confidently) in the expected language



| Language | Random search phrase |
|----------|--|
| English | <i>the northern territory</i> |
| Estonian | <i>ameerika ühendriikide relvajõud</i> |
| Finnish | <i>hüleen liittynyt hydroksyyliiryhmä</i> |
| German | <i>abgesetzten enameloliden zahnkappen</i> |
| Latvian | <i>starptautiskajā šaha turnīrā</i> |
| Russian | <i>совета рабочих депутатов</i> |
| Spanish | <i>administración del estado</i> |
| Urdu | <i>انہ اور فرانس</i> |

VoxLingua107: post-processing

- Many steps to filter out noise (i.e., data not in the expected language):
 - Language identification applied to retrieved video title and description
 - Speech/non-speech detection to filter out music
 - Speech segments were split into utterances of max 20 second duration
- A random subset of the data was manually verified by volunteers
 - 85% of utterances in the expected language
- Data-driven classifier was used to remove outliers
 - Result: 98% of utterances in the expected language



VoxLingua107: facts

| Language | h | Language | h | Language | h | Language | h | Language | h | Language | h | Language | h |
|-------------|-----|-----------|-----|-------------|----|-------------|-----|------------|-----|-----------|-----|----------------|-------------|
| Abkhazian | 10 | Catalan | 88 | German | 39 | Kannada | 46 | Marathi | 85 | Shona | 30 | Tibetan | 101 |
| Afrikaans | 108 | Cebuano | 6 | Greek | 66 | Kazakh | 78 | Mongolian | 71 | Sindhi | 84 | Turkish | 59 |
| Albanian | 71 | C. Khmer | 41 | Guarani | 2 | Korean | 77 | Nepali | 72 | Sinhala | 67 | Turkmen | 85 |
| Amharic | 81 | Chinese | 44 | Gujarati | 46 | Lao | 42 | Norwegian | 107 | Slovak | 40 | Ukrainian | 52 |
| Arabic | 59 | Croatian | 118 | Haitian | 96 | Latin | 67 | Nynorsk | 57 | Slovenian | 121 | Urdu | 42 |
| Armenian | 69 | Czech | 67 | Hausa | 93 | Latvian | 42 | Occitan | 15 | Somali | 103 | Uzbek | 45 |
| Assamese | 155 | Danish | 28 | Hawaiian | 12 | Lingala | 90 | Panjabi | 54 | Spanish | 39 | Vietnamese | 64 |
| Azerbaijani | 58 | Dutch | 40 | Hebrew | 96 | Lithuanian | 82 | Persian | 56 | Sundanese | 64 | Waray | 11 |
| Bashkir | 58 | English | 49 | Hindi | 81 | Luxembourg. | 75 | Polish | 80 | Swahili | 64 | Welsh | 76 |
| Basque | 29 | Esperanto | 10 | Hungarian | 73 | Macedonian | 112 | Portuguese | 64 | Swedish | 34 | Yiddish | 46 |
| Belarusian | 133 | Estonian | 38 | Icelandic | 92 | Malagasy | 109 | Pushto | 47 | Tagalog | 93 | Yoruba | 94 |
| Bengali | 55 | Faroese | 67 | Indonesian | 40 | Malay | 83 | Romanian | 65 | Tajik | 64 | Total | 6628 |
| Bosnian | 105 | Finnish | 33 | Interlingua | 3 | Malayalam | 47 | Russian | 73 | Tamil | 51 | Average | 62 |
| Breton | 44 | French | 67 | Italian | 51 | Maltese | 66 | Sanskrit | 15 | Tatar | 103 | | |
| Bulgarian | 50 | Galician | 72 | Japanese | 56 | Manx | 4 | Scots | 3 | Telugu | 77 | | |
| Burmese | 41 | Georgian | 98 | Javanese | 53 | Maori | 34 | Serbian | 50 | Thai | 61 | | |

VoxLingua107: results

- Result: 7.6% error rate
- Try it yourself:
<https://huggingface.co/speechbrain/lang-id-voxlingua107-ecapa>
- Data available here:
<http://bark.phon.ioc.ee/voxlingua107/>
- Pretrained model available here:
<https://huggingface.co/speechbrain/lang-id-voxlingua107-ecapa>



Urdu → Hindi

Spanish → Galician

Norwegian → Nynorsk

Dutch → Afrikaans

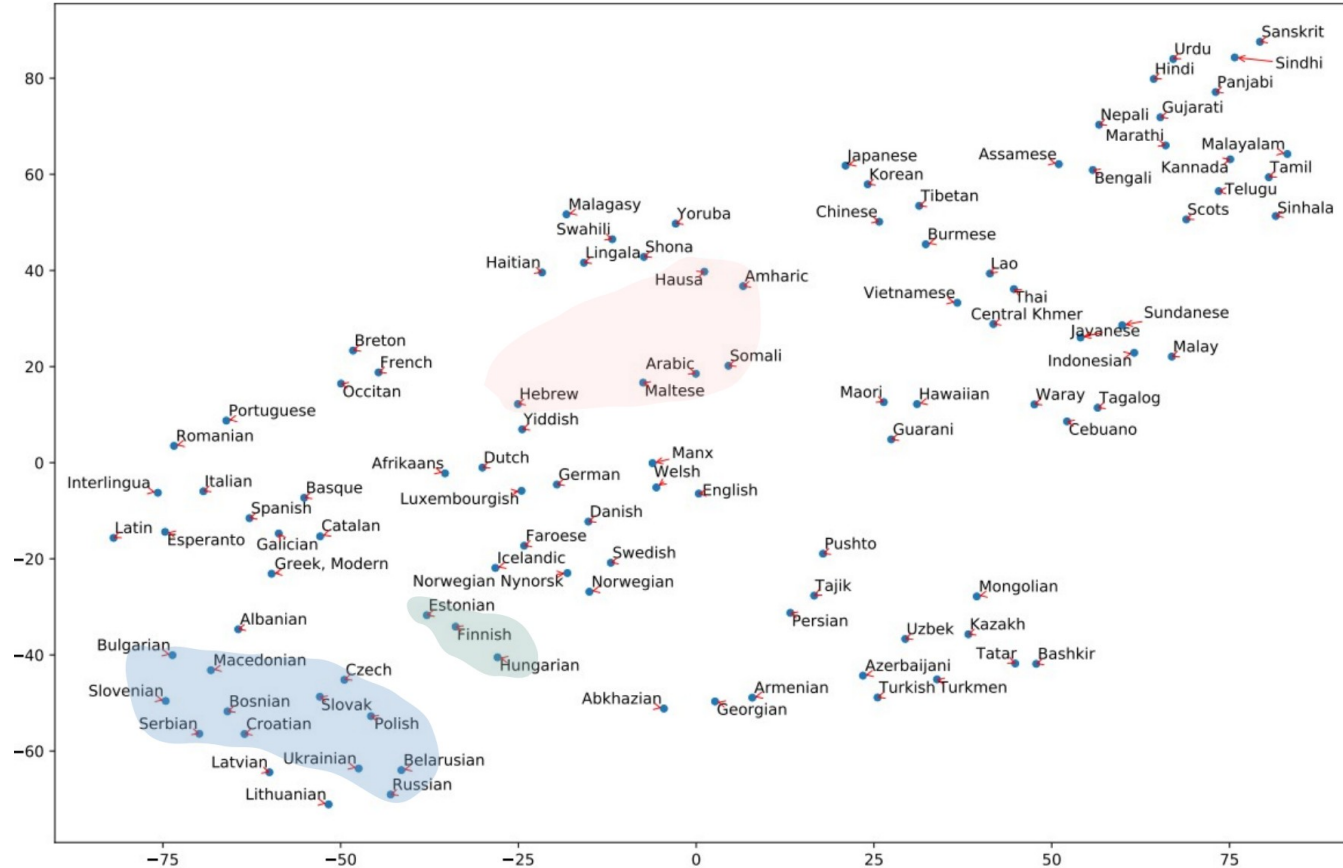
English → Welsh

Estonian → Finnish

Most common errors

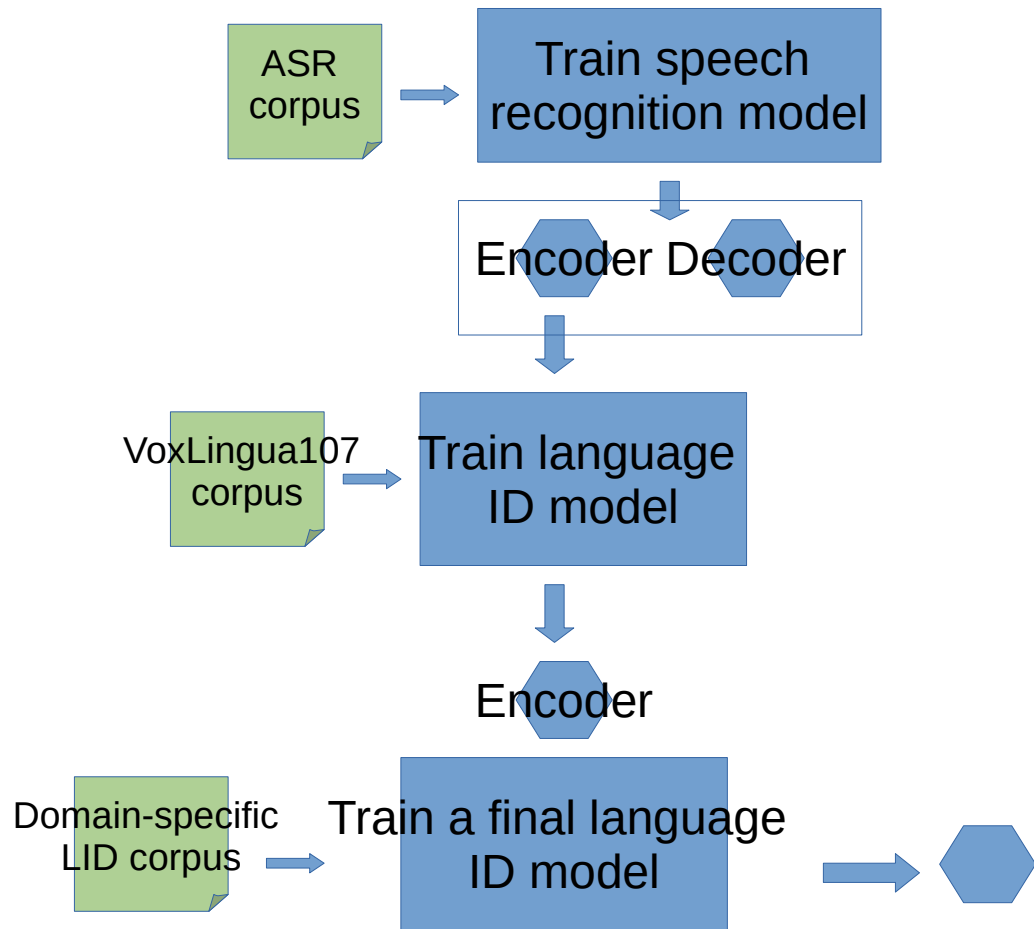
VoxLingua107: side effect

- t-SNE projection of language embeddings, averaged over language training data
- Reflects the true hierarchy of language families
- Some obvious deviations, likely due to geographical and cultural influences
- E.g., Slavic, Finno-Ugric and Afroasiatic groups



Language ID benefits from ASR

- Language identification system can be pretrained on a speech recognition task
- Pipeline:
 - Train an encode-decoder ASR model on some data (multilingual, or just a very large and diverse monolingual corpus)
 - Throw away decoder, use encoder as a finetunable feature extractor
 - Add attention-based pooling mechanism, train a language ID system on VoxLingua107
 - Result: very good feature extractor for (almost) any language ID task



Language ID benefits from ASR, II

- Task 1: closed-set language ID task with 6 languages (Cantonese, Korean, Indonesian, Vietnamese, Russian), training data 10h per language
- Task 2: noisy language ID task with 5 Asian languages (10h per language)
- Task 3: 11 South-African languages (Afrikaans, English, isiNdebele, isiXhosa, ..), 2 hours training data per language
- Table shows error rates

| | Task 1 | Task 2 | Task 3 |
|--|--------|--------|--------|
| Trained only on task training data | 11.9 | 6.8 | 26.4 |
| Pretrained on VoxLingua107 | 2.8 | 1.3 | 12.4 |
| + First pretrained on Mozilla CommonVoice (multilingual ASR) | 0.6 | 0.8 | 10.1 |
| + First pretrained on WeNetSpeech (10K hour Mandarin ASR corpus) | 0.8 | 0.4 | 9.7 |

Some general Tips & Tricks about Training neural Networks

Partly based on Andrej Karpathy's (Head of AI at Tesla at that time) blog post <http://karpathy.github.io/2019/04/25/recipe/> (must read)

Data

- Become one with data
 - Inspect your data carefully (spend many hours)
 - Understand the distribution and look at the patterns
 - Your brain is very good at it
 - Are there duplicates?
 - Are there corrupt examples?
 - Are the labels imbalanced?
 - Think how your brain solves this task
 - Can you solve this task on local context only?
 - Or do you need larger (global) context?

How much data?

- Very general rule: you can start thinking about training a DNN if you have at least 1000 training samples
- There is no data like more data
- Common situation:
 - Baseline accuracy with a simple model: accuracy 60%
 - Super-well tuned “transformer with multi-head attention”: accuracy: 70%
 - Spend 1 week
 - Simple model with more data: 80%
 - Spend 2 days on data collection
- Spending time on data collection is a time well spent
 - Can always apply new models / training tricks when they are invented
 - Time spent on fine-tuning a fancy model might be wasted when suddenly a new model architecture is invented

Data augmentation

- *“The next best thing to real data is half-fake data”*
- Data augmentation: train artificial additional training data by corrupting the real data by some domain-specific way
- Often applied so that the amount of training data is increased 10 (or more) times
- Data augmentation can make a huge difference, especially if domain adaptation can be performed using augmentation

Data augmentation, continued

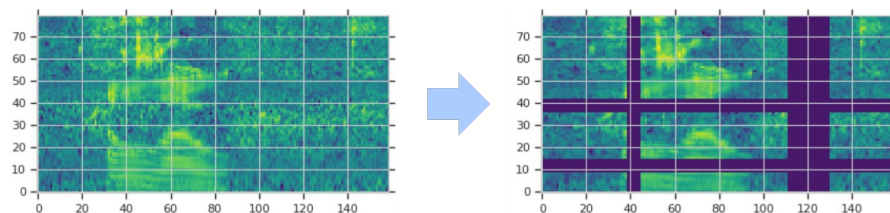
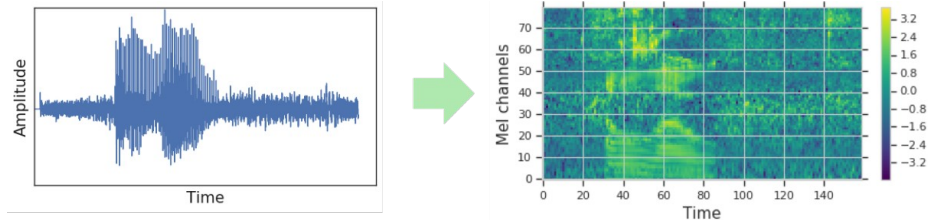
- For example:
 - Speech recognition model trained on clean data: accuracy on reverberant/noisy data **50%**
 - Add noise and reverberate clean training data: accuracy **85%**
- Data augmentation is more useful in domains where input data is “continuous”
 - Image processing
 - Speech processing
- Usually works better than trying to “fix” the test data (e.g., denoise)
- Almost never hurts accuracy

Data augmentation is speech recognition

- Speed perturbation
 - Add copies of training sentences, sped up or slow down by 10%
 - Simulates faster/slower speaking, but also different vocal tract characteristics
- Reverberation
 - Add room effects (using real or artificial **impulse responses**)
- Noise augmentation
 - Add background noise to (clean training data), e.g. street noise, engine, car, wind
 - Add “foreground noises” (e.g., door slams, claps, clicks)
 - Add music
 - Add “babble noise” (many persons speaking in the background)
 - All this with random signal-to-noise ratio, and many times

Spectral augmentation

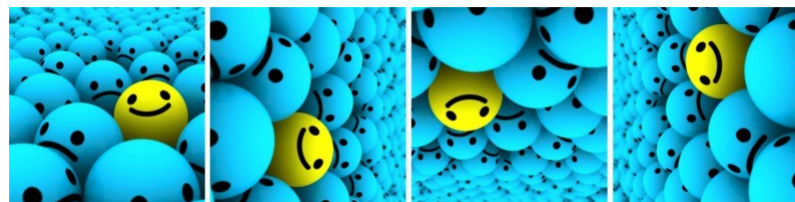
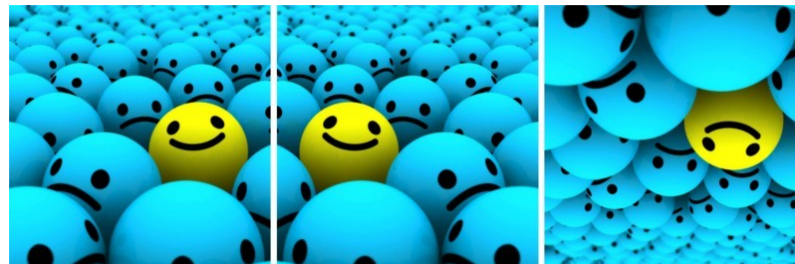
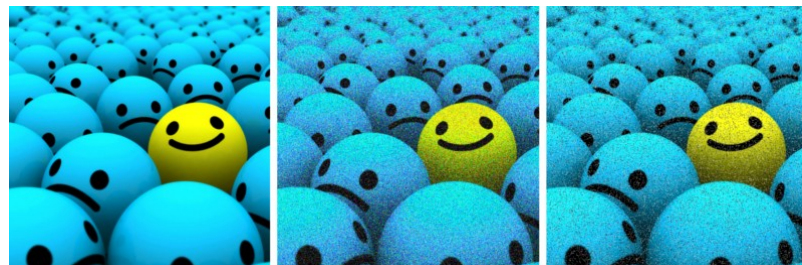
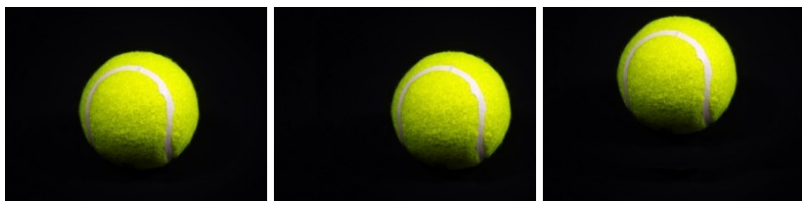
- Proposed by Google researchers
<https://ai.googleblog.com/2019/04/specaugmentation-new-data-augmentation.html>
- Three types of modifications of the raw filterbank spectrogram, randomly chosen
 - Time warping
 - Masking blocks of consecutive filterbank features (horizontal)
 - Masking whole filterbank blocks in time (vertical)



| | LibriSpeech 960h | | Switchboard 300h | |
|---------------|------------------|------------|------------------|-------------|
| | test-clean | test-other | Switchboard | CallHome |
| Previous SOTA | 2.95 | 7.50 | 8.3 | 17.3 |
| Our Results | 2.5 | 5.8 | 6.8 | 14.1 |

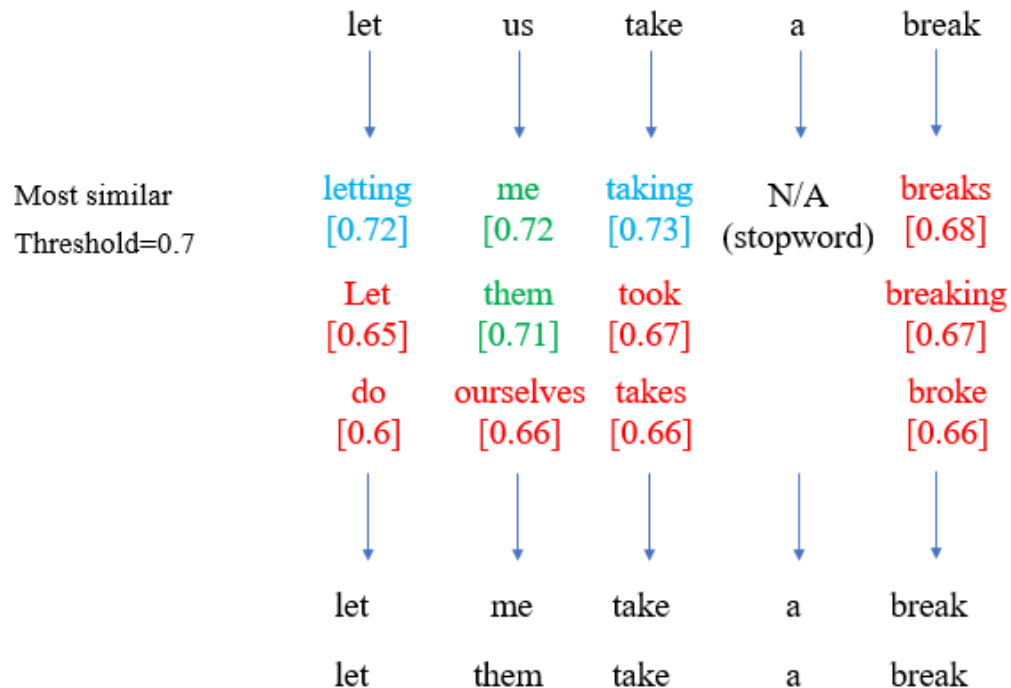
Data augmentation with images

- Flipping
- Rotation
- Translation
- Scaling
- Adding noise
- ...



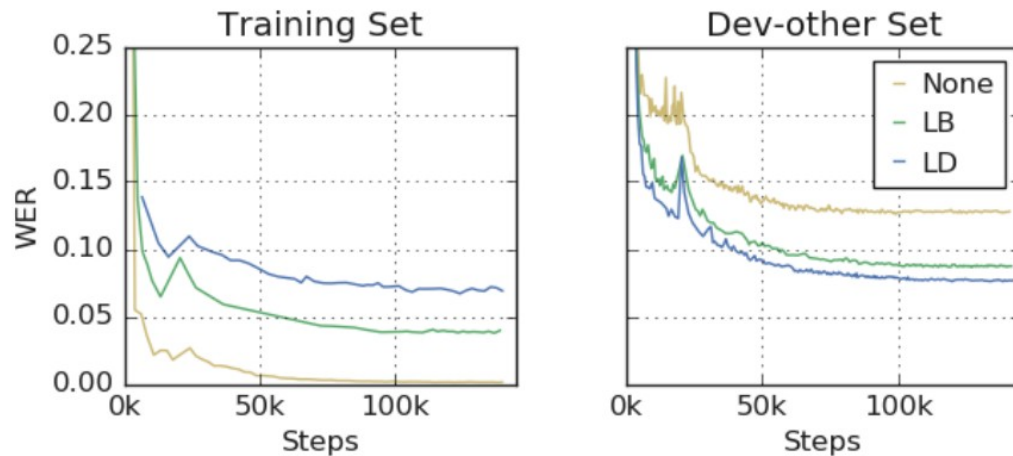
Data augmentation for NLP

- For text data, data augmentation is less useful
- One idea: replace words in training data with words that have similar embeddings
 - But usually better to just use pretrained word embeddings (or pretrained contextualized representations, like BERT)
- Alternative: use domain knowledge
 - E.g., replace company names in text with other company names from the business registry (if having a good coverage of company names is relevant to your task)
- Or: use machine translation and backtranslation:
 - Original:
The restaurant serves huge breakfast for a very cheap price.
 - Translated to German and back:
The restaurant serves a huge breakfast at a very reasonable price.
 - Translated to Japanese and back:
A rich breakfast is served at the restaurant at an affordable price.



Data augmentation under-fits training data

- Data augmentation turns over-fitting problem into under-fitting problem
 - Loss when training augmented data is worse than when training on clean data
 - But loss (and error rate) on test data is better
- Thus, common methods that work against under-fitting help:
 - Using larger (wider and deeper) models
 - Training for more epochs



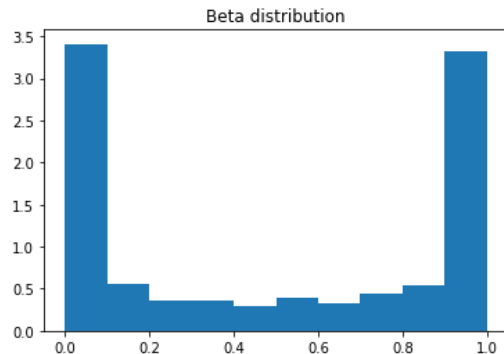
Mixup

- During training, mix two training samples and the corresponding labels

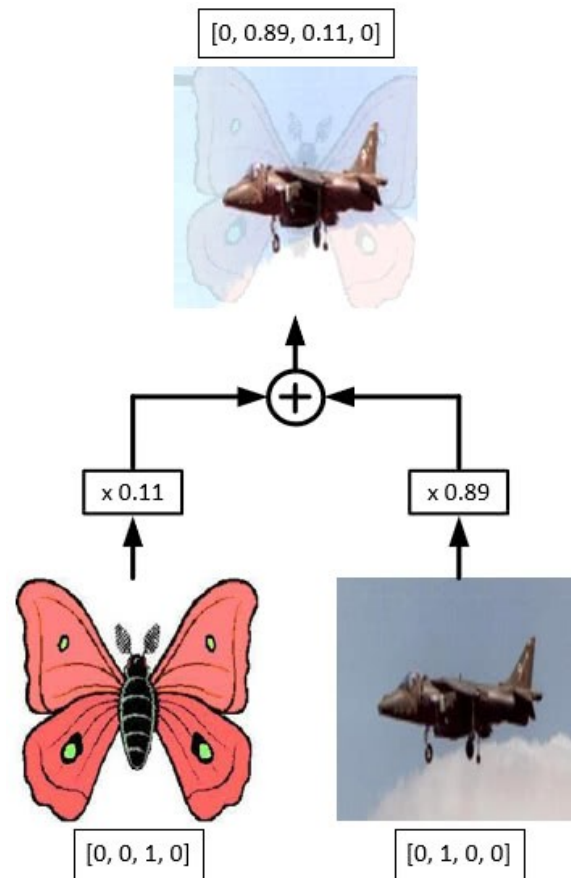
$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j, \quad \text{where } x_i, x_j \text{ are raw input vectors}$$

$$\tilde{y} = \lambda y_i + (1 - \lambda)y_j, \quad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

- The mixup factor is drawn randomly from a beta distribution



- Kind of surprising that it helps
- Not really applicable to text data

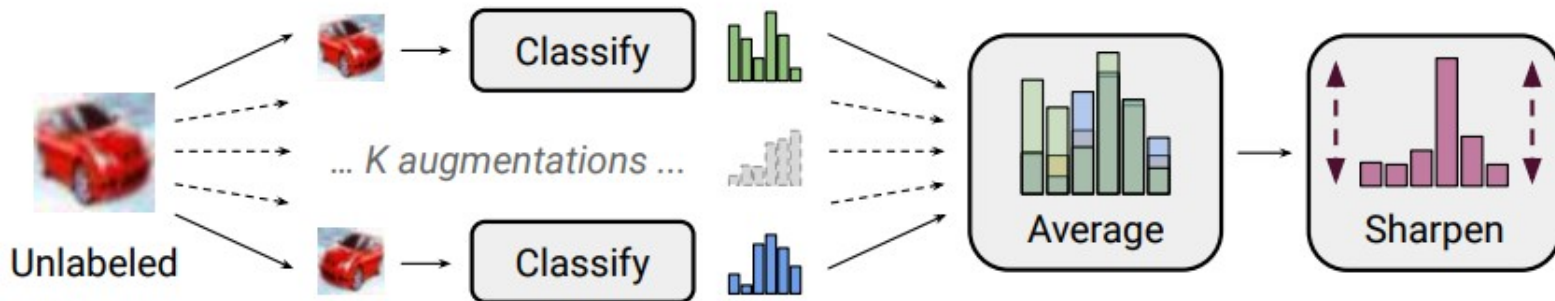


MixMatch: semi-supervised training

Given: some **labeled data** and a lot of **unlabeled data**

Procedure:

- Train a first model on labeled data (with data augmentation)
- Generate pseudo-labels for unlabeled data
 - This is done by classifying different augmentations of the unlabeled sample, averaging the predictions and sharpening the resulting probability distribution (to make it less smooth and more one-hot-like), see below
- Train a new model, using mixup between labeled and pseudo-labeled samples



Pretraining

- *It rarely ever hurts to use a pretrained network if you can, even if you have enough data*
- Pretraining was first popularized by the image recognition community
 - Pretrain a ConvNet on ImageNet
 - ImageNet: dataset of 1.2M images, 1000 categories
 - Models trained on ImageNet can be used to initialize models **for completely other datasets** and improve performance significantly
 - Works even if you have only a few examples per object

Image classification

Easiest classes

red fox (100) hen-of-the-woods (100) ibex (100) goldfinch (100) flat-coated retriever (100)



tiger (100)



hamster (100)



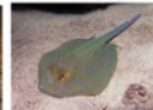
porcupine (100)



stingray (100)



Blenheim spaniel (100)



Hardest classes

muzzle (71) hatchet (68) water bottle (68) velvet (68) loupe (66)



hook (66)



spotlight (66)



ladle (65)



restaurant (64)



letter opener (59)



What model to use in NLP?

- What model to use for NLP task (e.g. text classification, information extraction), given that you have reasonable amount of training data?
- For most NLP tasks, **Transformer** (multi-layer multi-head attention with position encodings) is currently the best choice
- For English, use BERT (or some of its descendants, like XLM-Roberta) for getting contextualized word embeddings
- For Estonian, try XML-Roberta and EstBERT
- Consider machine translation, if you can afford
 - Many hard tasks work much better in English, due to abundance of training data
 - For example, abstractive summarization: translate Estonian test data to English, apply an English summarization model, and translate results back to Estonian
 - Works better than an Estonian model trained on little data, despite MT errors



Aran Komatsuzaki

@arankomatsuzaki

Follow



Whenever I hear the name 'LSTM,' I can't help but wonder if the person is still living in pre-2017.

6:13 AM - 28 Apr 2019

1 Like



Debugging neural networks

- *Neural net training fails silently*
- Tricky to unit test
- Your net can still (shockingly) work pretty well even if your training data is corrupted somehow
 - E.g., maybe you feed it columns instead of rows of data
 - It's because neural nets can memorize the data to some extent
- Most of the time when you screw something up, it will train but silently work a bit worse

Recipe for training neural nets

- Verify your data
- Start with simple models
 - No data augmentation, fancy learning rate schedules, etc
- Train an input-independent baseline, (e.g. easiest is to just set all your inputs to zero)
 - This should perform worse than when you actually plug in your data without zeroing it out. Does it?
- Overfit a single batch of only a few examples (e.g. as little as two).
 - Verify that we can reach the lowest achievable loss (e.g. zero)

Recipe, continued

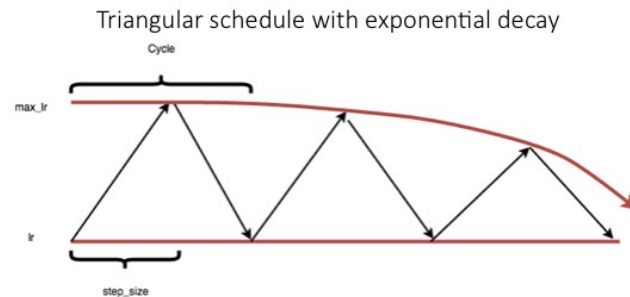
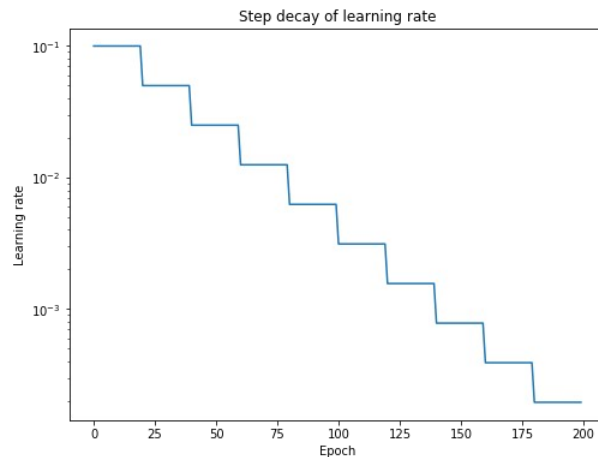
- Don't be a hero
 - Start with a model that is known to give good results on similar tasks
 - You're allowed to do something more custom later and beat this
- One complexity at a time
 - At fanciness to your model one by one and make sure it improves the model
- Leave learning rate tuning to the end

Recipe, continued

- Regularize (avoid overfitting)
 - Try to get more data
 - Data augmentation, try to be creative
 - Pretraining rarely hurts
 - Semisupervised learning usually gives very little improvement, and is very complicated
 - But recently big progress
 - Try with less features
 - Try smaller model size
 - Add dropout
 - Add L2 regularization (weight decay)
 - Early stopping
 - Try larger model, but stop early

Some tricks: learning rate schedule

- Learning rate schedule
 - E.g., step decay (0.1 at the beginning, 0.0001 at the end)
 - Or: constant learning rate, but decrease it when the loss on dev data doesn't decrease any more (ReduceLRonPlateau in Pytorch)
 - More recent: triangular learning rates
 - Start at learning rate 0.001
 - Gradually increase it to 0.01 at epoch 25%
 - Then gradually decrease it until the end
 - Cyclic learning rates



Some tricks: dropout, label smoothing

- Similarly to learning rate, dropout can be also scheduled
 - E.g.: 0 dropout at the beginning,
 - Increase dropout gradually to 0.3 at epoch 50%
 - Then gradually decrease it back to 0
- Label smoothing
 - Very simple idea: redistribute 0.1 of probability mass from the correct label to all other labels
 - Often used in Google's papers
 - Results in less loss fluctuations on dev data during epochs in my experience
 - Maybe disable label smoothing during the last training phase

Ensembles

- *Model ensembles are a pretty much guaranteed way to gain 2% of accuracy on anything*
 - Ensembles of models of different architecture result in the best performance
 - Even if a particular model is relatively bad (compared to the best ones in the ensemble), combining its predictions with others can give surprising gains
- *If you can't afford the computation at test time, try distilling your ensemble into a network using knowledge distillation*
 - Compute posteriors (target probabilities) for your training data using the ensemble, and train a final model on the posteriors (instead of real targets)
 - This way, the final model learns to mimic the ensemble