

- A landscape PDF version
- A breakdown of “when to choose CNN vs ViT in 2025”
- Modern CNN architectures you should know (ConvNeXt, EfficientNetV2, YOLO11, etc.)

Just tell me.



what is the ppl measure reported by d2l recurrent nnets?

PPL = *Perplexity* — a standard evaluation metric for language models, including the RNN-based language models in **D2L (Dive Into Deep Learning)**.

Here is the clean definition and how D2L computes it.

✅ What does PPL (Perplexity) mean?

Perplexity measures how well a model predicts a sequence of tokens.

Lower perplexity → better model quality.

Formally:

$$\text{Perplexity} = \exp \left(-\frac{1}{N} \sum_{t=1}^N \log p(x_t \mid x_{<t}) \right)$$

This is simply:

Per-token average negative log-likelihood, exponentiated.

🎯 Intuition

If the model is “perplexed” by the data, perplexity is high.

If the model predicts the next token well, perplexity is low.

Interpretation:

- PPL = 1 ⇒ perfect model
- PPL = vocabulary size ⇒ model is guessing uniformly
 - e.g. vocab=10k, perplexity≈10000 means “no clue”

- LSTM/GRU models usually get PPL 30–60 on small datasets
 - GPT-type models on large corpora: PPL < 10
-

How D2L specifically reports perplexity

In the **D2L RNN chapters**, they compute:

```
ini
```

[Copy code](#)

```
perplexity = math.exp(loss)
```

Why?

Because in language modeling, the cross-entropy loss (in nats, not bits) over one token is:

$$\ell = -\log p(x_t)$$

So the average loss is:

$$\text{loss} = -\frac{1}{N} \sum \log p(x_t)$$

Then:

$$\text{PPL} = e^{\text{loss}}$$

D2L does exactly that.

Concrete Example (straight from D2L code structure)

In the training loop, after each epoch:

```
python
```

[Copy code](#)

```
ppl = math.exp(l)
```

Where `l` is the average loss per token.

So:

- If average loss per token = 3.0 → PPL = $e^3 \approx 20.1$
- If loss improves to 2.5 → PPL = $e^{2.5} \approx 12.2$

This is the number printed by `d2l.predict_rnn` and `d2l.train_epoch_ch8`.

⚠ Important detail

D2L computes perplexity **per token**, not per sequence.

This is the standard metric for language models.

✓ TL;DR

Perplexity (PPL) in D2L = $\exp(\text{average token cross-entropy})$

Meaning: expected branching factor / effective vocabulary size.

Lower = better.

If you want, I can also show:

- How perplexity relates to bits-per-token
- How to compute perplexity on batches in PyTorch
- How transformer models report perplexity
- Common PPL ranges for RNN vs GPT models

Just ask.



is there a measure used "perplexity per sequence"?

Yes — you *can* define “perplexity per sequence,” but it’s *not* commonly used, for several reasons.

Let me explain both the *theoretical definition* and *why it’s rarely reported* in practice.

✓ 1. Standard Perplexity (the one used in RNNs, Transformers, D2L)

Traditional perplexity is computed **per token**, not per sequence:

$$\text{PPL} = e^{\text{average token cross-entropy}}$$

This answers:

"On average, how many equally likely choices does the model face at each step?"

This is why the metric is useful: **it is length-normalized.**

✅ 2. What would "Perplexity Per Sequence" mean?

If you define total sequence log-likelihood:

$$\text{NLL}_{\text{sequence}} = - \sum_{t=1}^T \log p(x_t \mid x_{<t})$$

Then sequence perplexity would be:

$$\text{PPL}_{\text{sequence}} = e^{\text{NLL}_{\text{sequence}}}$$

This is equivalent to:

$$\text{PPL}_{\text{sequence}} = \prod_{t=1}^T \frac{1}{p(x_t \mid x_{<t})}$$

Basically: **the reciprocal probability of the entire sequence.**

⚠️ Why this is almost never used

❌ (1) It grows exponentially with sequence length

If the average per-token perplexity is ~20 and the sequence has 50 tokens:

$$20^{50} \approx 10^{65}$$

This number is meaningless and impossible to compare across sequences.

❌ (2) It strongly depends on sequence length

Two identical models get different "sequence perplexity" values for:

- different sequence lengths
- different datasets
- different tokenizations

So it's **not comparable** and therefore not useful.

✗ (3) It hides what the model struggles with

A single badly predicted token makes the entire sequence perplexity explode.

Per-token PPL gives **much more granular** insight.

✓ 3. Useful variants that *are* sometimes used

(a) Perplexity per sequence *averaged across sequences*

Sometimes people report something like:

$$\text{PPL}_{\text{avg-seq}} = \exp \left(\frac{1}{M} \sum_{\text{sequences}} \frac{1}{T_i} \sum_{t=1}^{T_i} -\log p(x_t) \right)$$

This is still **per token**, just aggregated per sequence.

(b) Bits-per-byte / bits-per-character

Used in:

- OpenAI GPT-2 evals
- Google's text models
- Hutter prize datasets
- Character-level LM benchmarking

This *is* a per-sequence measure but normalized by length → comparable.

★ 4. The closest meaningful thing to "sequence perplexity"

People sometimes measure:

Sequence Negative Log-Likelihood (SeqNLL)

$$\text{SeqNLL} = - \sum_{t=1}^T \log p(x_t)$$

or:

Average log-likelihood over sequences

$$\frac{1}{M} \sum_{i=1}^M \text{SeqNLL}(x^{(i)})$$

But they rarely exponentiate it.

🎯 5. Practical Answer

✓ Yes, you *can* define perplexity for an entire sequence.

✗ But nobody uses it.

✓ Instead, we use token-level perplexity because it's normalized and comparable.

If you want, I can show you:

- how to compute sequence-level perplexity in PyTorch
- how to compute per-token vs per-sequence NLL
- a comparison of how perplexity scales with sequence length
- what the metric looks like on your own dataset

Just say the word.



what are other measures used for RNN-seq models?

RNN-based sequence models (language models, sequence predictors, time-series models, etc.) use a *variety* of evaluation measures — **perplexity** is only the most famous one for text.

Here's a complete, domain-organized list used today (2025), grouped by *what type of sequence problem* the RNN is solving.

🔵 1. If the RNN is a *language model* (predicting next token)

Primary metrics

Metric	Why Used
Perplexity (PPL)	Standard for LM quality; token-level
Cross-Entropy Loss	Direct training loss; easier for optimization research
Bits-per-Character / Bits-per-Byte	Used in char-level models, compression-like tasks
Negative Log-Likelihood (NLL)	Useful without exponentiation ($PPL = \exp(NLL)$)

Relevance in 2025

Transformers displaced RNNs, but RNNs and SSMs (Mamba, S4) still use the same metrics.

2. If the RNN is used for *sequence classification*

(e.g., sentiment classification, spam detection, ECG classification)

Metrics

- Accuracy
- Precision, Recall
- F1-score
- AUROC (common for medical datasets)
- AUPRC (imbalanced label sets)

No perplexity here — because you're not predicting tokens.

3. If the RNN is predicting *labels at each time step*

(e.g., POS tagging, NER, frame labeling, speech phoneme labeling)

Metrics

- Token accuracy
- Sequence accuracy
(all tokens in a sequence must match)
- Edit distance (Levenshtein) for approximate alignment
- F1 score (for chunking tasks like NER)

4. If the RNN is used for *sequence-to-sequence tasks*

(e.g., translation, summarization, dialogue — classical seq2seq)

Metrics used:

- **BLEU**
(classic for machine translation)
- **ROUGE**
(summarization)
- **CIDEr**
(captioning)
- **METEOR**
- **CHRF++**
- **BERTScore** (neural similarity)
- **MoverScore**
- **COMET** (neural MT metric, SOTA)

These are *not RNN-specific* — just standard seq2seq metrics.

5. If the RNN is used for *time-series forecasting*

(e.g., stock prediction, sensor prediction, weather)

Metrics

- **MAE**
- **RMSE**
- **MAPE**
- **SMAPE**
- **R²**
- **CRPS** (for probabilistic forecasts)
- **Pinball loss** (quantile forecasting)
- **NLL** for probabilistic RNNs (e.g., Gaussian output head)

These depend on the output distribution.

6. If the RNN is used in *reinforcement learning (RL)*

(e.g., recurrent PPO, A3C with LSTM)

RNNs store hidden state across time steps.

Metrics

- **Episode reward / average return**
- **Sample efficiency** (reward vs steps)
- **Policy entropy**
- **Success rate** (in robotics tasks)

No perplexity here either.

7. If the RNN is used in *speech processing*

(before transformers dominated)

For ASR:

- **WER (Word Error Rate)**
- **CER (Character Error Rate)**

For speech tagging:

- same as above (frame accuracy, F1).
-

8. If it's a *generative model* (music, handwriting, motion, etc.)

Measures:

- **NLL / PPL** (if discrete)
 - **Mean squared error** (if continuous motions)
 - **Fréchet Audio Distance (FAD)** (speech/music)
 - **Fréchet Video Distance (FVD)** (video generation)
 - **Dynamic Time Warping (DTW)** (gesture/trajectory similarity)
-

Quick Summary

RNN sequence models can be evaluated by:

1. Language modeling

- Perplexity
- Cross-entropy

- Bits-per-char/byte

2. Classification (sequence or token)

- Accuracy
- F1
- AUROC

3. Seq2seq tasks

- BLEU / ROUGE / METEOR / COMET

4. Time-series

- MAE, RMSE, MAPE, CRPS, Pinball loss

5. Speech

- WER, CER

6. RL

- Rewards, success rate

7. Generative models

- NLL
- FAD / FVD
- DTW

If you want, I can create:

- a table comparing all these metrics
- a landscape PDF
- PyTorch examples of computing each metric
- a flowchart: "Which metric to use for your RNN task?"

Just tell me.



can you make a table, including the defintion in math,
and definition in english, explain use cases etc

Below is a **clean, comprehensive table** of the *main evaluation metrics for RNN-based sequence models*, including:

- **Mathematical definition**
- **Plain-English definition**
- **Use cases**
- **Notes / comments**

I organized it by task type (LM, seq2seq, classification, time-series, speech, RL).

Master Table of RNN Sequence-Model Metrics

Legend

- x_t = ground-truth token/value at time t
- \hat{x}_t = model prediction
- T = sequence length
- $p(x_t \mid x_{<t})$ = predicted probability of next token

1. Language Modeling Metrics (next-token prediction)

Metric	Math Definition	Plain-English Meaning	Used For
Cross-Entropy Loss	$CE = -\frac{1}{T} \sum_{t=1}^T \log p(x_t)$	Average surprise the model has when predicting the next token. Lower = better.	Training loss for RNN/GRU/LSTM/Transformer LMs.
Perplexity (PPL)	$PPL = \exp(CE)$	“Effective branching factor.” On average, how many equally likely choices the model faces.	Standard eval metric for language models.
Sequence Negative Log-Likelihood (SeqNLL)	$SeqNLL = -\sum_{t=1}^T \log p(x_t)$	Raw un-normalized log-loss over the whole sequence.	Comparing model likelihoods over full sentences.

Metric	Math Definition	Plain-English Meaning	Used For
Bits-per-Character (BPC)	$\text{BPC} = \frac{1}{T} \sum_{t=1}^T \log_2 \frac{1}{p(x_t)}$	Average number of bits needed per character. Comparable across tokenizations.	Character-level models, compression-style evals.

2. Sequence Classification Metrics (one label per full sequence)

Metric	Math Definition	Plain-English Meaning	Used For
Accuracy	$\frac{\text{\#correct}}{\text{\#total}}$	Fraction of sequences classified correctly.	Sentiment classification, spam detection.
Precision	$\frac{TP}{TP + FP}$	When the model predicts positive, how often it's correct.	Imbalanced classes.
Recall	$\frac{TP}{TP + FN}$	Out of actual positives, how many the model found.	Medical/rare event detection.
F1 Score	$2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$	Harmonic mean of precision/recall.	NER, sequence classification with imbalance.
AUROC	Probability model ranks a random positive above a random negative.	Measures ability to rank examples.	Medical datasets, binary tasks.
AUPRC	Area under Precision–Recall curve	Better for highly imbalanced labels.	Rare-event tasks.

3. Token-Level Sequence Labeling Metrics (POS, NER, frame tagging)

Metric	Math Definition	Plain-English Meaning	Used For
Token Accuracy	$\frac{\sum_t \mathbf{1}[\hat{x}_t = x_t]}{T}$	Fraction of tokens labeled correctly.	POS tagging, morphological tagging.
Sequence Accuracy	$\mathbf{1}[\forall t : \hat{x}_t = x_t]$	Whether the entire sequence is correct. All-or-nothing.	Strict sequence labeling.
F1 (for spans/chunks)	F1 on predicted vs true entity spans.	Quality of predicted spans/chunks, not individual tokens.	NER, chunking.
Edit Distance / Levenshtein	Minimum operations to convert prediction → ground truth.	Measures similarity of two sequences allowing insert/delete.	Speech alignment, OCR, handwriting.

4. Seq2Seq Generation Metrics (translation, summarization, captioning)

Metric	Math Definition	Plain-English Meaning	Used For
BLEU	n-gram precision with brevity penalty.	Measures how many n-grams overlap with reference.	Machine translation (classic).
ROUGE	Overlap of recall-oriented n-grams (ROUGE-1/2/L).	Measures how much of the reference is recovered.	Summarization.
METEOR	Exact match + stem + synonym + paraphrase scoring.	More semantic than BLEU.	MT (older, robust).
CIDEr	TF-IDF weighted n-gram similarity.	Captioning-specific relevance measure.	Image captioning tasks.
BERTScore	Cosine similarity of contextual embeddings.	Measures semantic similarity using BERT.	Modern generation eval.
COMET	Learned neural metric predicting human judgment.	Most accurate automated MT metric (2024–2025).	MT evaluation.

5. Time-Series Forecasting Metrics (RNN for regression)

Metric	Math Definition	Plain-English Meaning	Used For
MAE	$\frac{1}{T} \sum_t x_t - \hat{x}_t $	Average absolute error.	General regression tasks.
RMSE	$\sqrt{\frac{1}{T} \sum_t (x_t - \hat{x}_t)^2}$	Penalizes big errors more than MAE.	Weather, sensor data.
MAPE	$\frac{100}{T} \sum_t \frac{ x_t - \hat{x}_t }{ x_t }$	Average percentage error.	Regression tasks where scale matters.
SMAPE	$\frac{100}{T} \sum_t \frac{ x_t - \hat{x}_t }{\max(x_t , \hat{x}_t)}$	Balanced version of MAPE.	Regression tasks with varying scales.
CRPS	$\int_{-\infty}^{\infty} (F_{\hat{x}}(z) - \mathbf{1}\{x \leq z\})^2 dz$	Measures accuracy of probabilistic forecasts.	Weather, probabilistic RNNs.
Pinball Loss	$L_{\alpha} = \max(\alpha(x - \hat{x}), (\alpha - 1)(x - \hat{x}))$	Quantile forecasting loss.	Demand prediction.

6. Speech and Sequence Alignment Metrics

Metric	Math Definition	Plain-English Meaning	Used For
WER	$\frac{S + D + I}{N}$ (substitutions, deletions, insertions)	How many word errors per output.	ASR (speech recognition).
CER	Same as WER but for characters.	Useful for languages w/ large vocabs (Chinese).	ASR, OCR.

7. Reinforcement Learning With RNN Policies

Metric	Math Definition	Plain-English Meaning	Used For
Average Return	$\mathbb{E}[\sum_t r_t]$	Mean cumulative reward per episode.	RL training curves.

Metric	Math Definition	Plain-English Meaning	Used For
Success Rate	fraction of successful episodes	Percentage of goals reached.	Robotics, navigation.
Policy Entropy	$-\sum_a \pi(a) \log \pi(a)$	How exploratory the policy is.	Stability / exploration analysis.

Summary (What to use for which RNN task)

RNN Task	Best Metrics
Language modeling	PPL, CE, BPC
Sequence classification	Accuracy, F1, AUROC