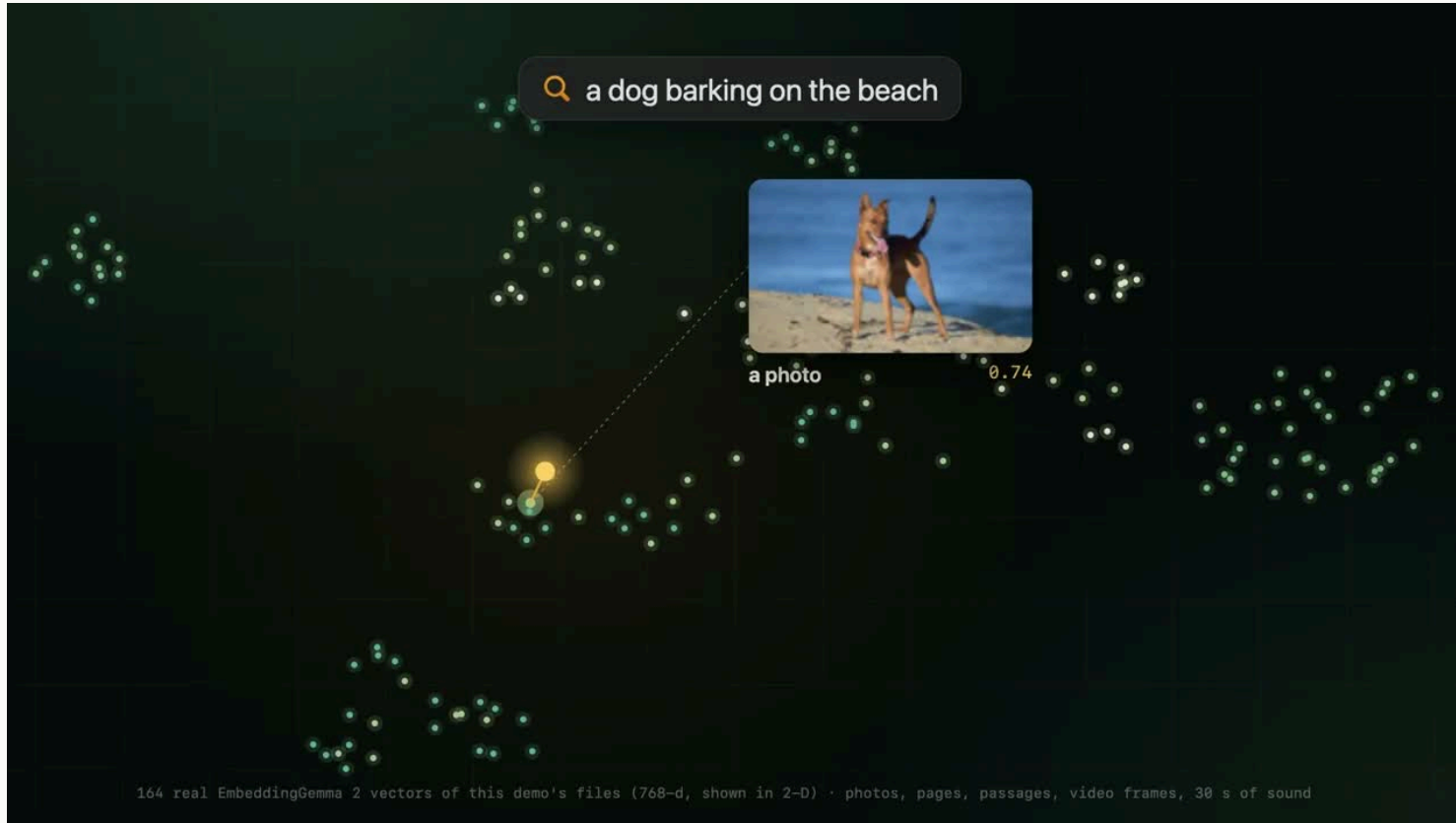


Can we find the moment, not just the file?

AUTHOR DEMO · DigUp · 10 October 2026



[Watch the original X demo](#) · Abdur Rahim · 60 s

DigUp turns a shared embedding space into local file search.

What do you remember?

“zebra in a video”

Open the matching moment.

“where they talk about sleep”

Open the audio segment.

“the clause about pets in the lease”

Open the PDF page.

The index stores pieces, vectors and locations

SOURCE-GROUNDED SCHEMATIC · vectors are not model outputs shown here

Photo

the picture

PDF

page / passage

Recording

overlapping 30 s windows

Video

sampled frames + audio



SQLite index

Piece	Vector	Location
video frame	[... 768 values ...]	file + time
PDF page	[... 768 values ...]	file + page
audio window	[... 768 values ...]	file + time

First pass: read the chosen folders. After an edit: refresh the affected entries.

An embedding says what a piece is about. Its metadata says where it came from.

Build once: extract → encode → save

TEACHING IMPLEMENTATION · extraction and media encoding are explicit adapter steps

```
pieces = extract(files) # text/pages/frames/audio
X = encode_pieces(pieces) # (N, 768)
X = unit_rows(X) # norm 1 per row
np.save("vectors.npy", X)
save_json("locations.json", locations)
```

Keep row order identical in the vector file and metadata file.

One row of metadata per vector

```
{"file": "lecture.pdf",
 "page": 42,
 "piece_id": "lecture-page-42",
 "model": "google/embeddinggemma-2",
 "revision": "PINNED_REVISION"}
```

Start with NumPy + JSON. A database becomes useful as the collection grows.

The dot product you already know does the search

ILLUSTRATIVE • 3D unit vectors for arithmetic; DigUp uses 768D

“zebra in a video”

→ text encoder → normalize

$q = [0.6, 0.8, 0]$

Stored piece	coordinate 1	coordinate 2	coordinate 3	$q \cdot v$
wildlife.mp4 · 00:24	0.8	0.6	0	0.96
lease.pdf · page 2	0	0.6	0.8	0.48
notes.txt · passage 3	0	0	1	0.00

$$0.6 \times 0.8 + 0.8 \times 0.6 + 0 \times 0 = \mathbf{0.96}$$

In the app: an $N \times 768$ matrix times a 768-dimensional query → N cosine scores.

The file vectors are already stored. Only the new query needs encoding.

Meaning helps. Exact words still matter.

IMPLEMENTATION DETAIL · DigUp revision f9b8106 · no measured ranking shown

Find the invoice containing **4815162342**

Semantic route

Query embedding → cosine scores

Find related content

Calibrate across modalities;
keep each file's best segment.

Word route

Filename + text + OCR

Look for the actual identifier

An embedding need not preserve
every digit of an invoice number.

Semantic evidence + word coverage + file-type hint → **ranked files**

DigUp combines learned similarity with ordinary search rules.

At query time: encode → score → look up the source

TEACHING IMPLEMENTATION · semantic lookup baseline, before hybrid ranking

```
X = np.load("vectors.npy")           # (N, 768), already normalized
q = encode_query(question)           # (768,)
q = q / np.linalg.norm(q)
scores = X @ q                       # (N,), one cosine per piece
best = np.argsort(-scores)[:5]
for row in best:
    print(scores[row], locations[row])
```

No re-encoding the whole collection for every question.

Edit a file → refresh its pieces. Change the embedding model → rebuild the compatible index.

For a small course collection, exact matrix multiplication is enough to begin.

A search result needs a route back to its source

SCHEMATIC · example queries from the project; course exercise is an analogy

Your query	Winning indexed piece	Open here
"pets in the lease"	PDF page / text passage	PDF page
"zebra in a video"	sampled video frame	timestamp
"code: retry with backoff"	source-code segment	file + line

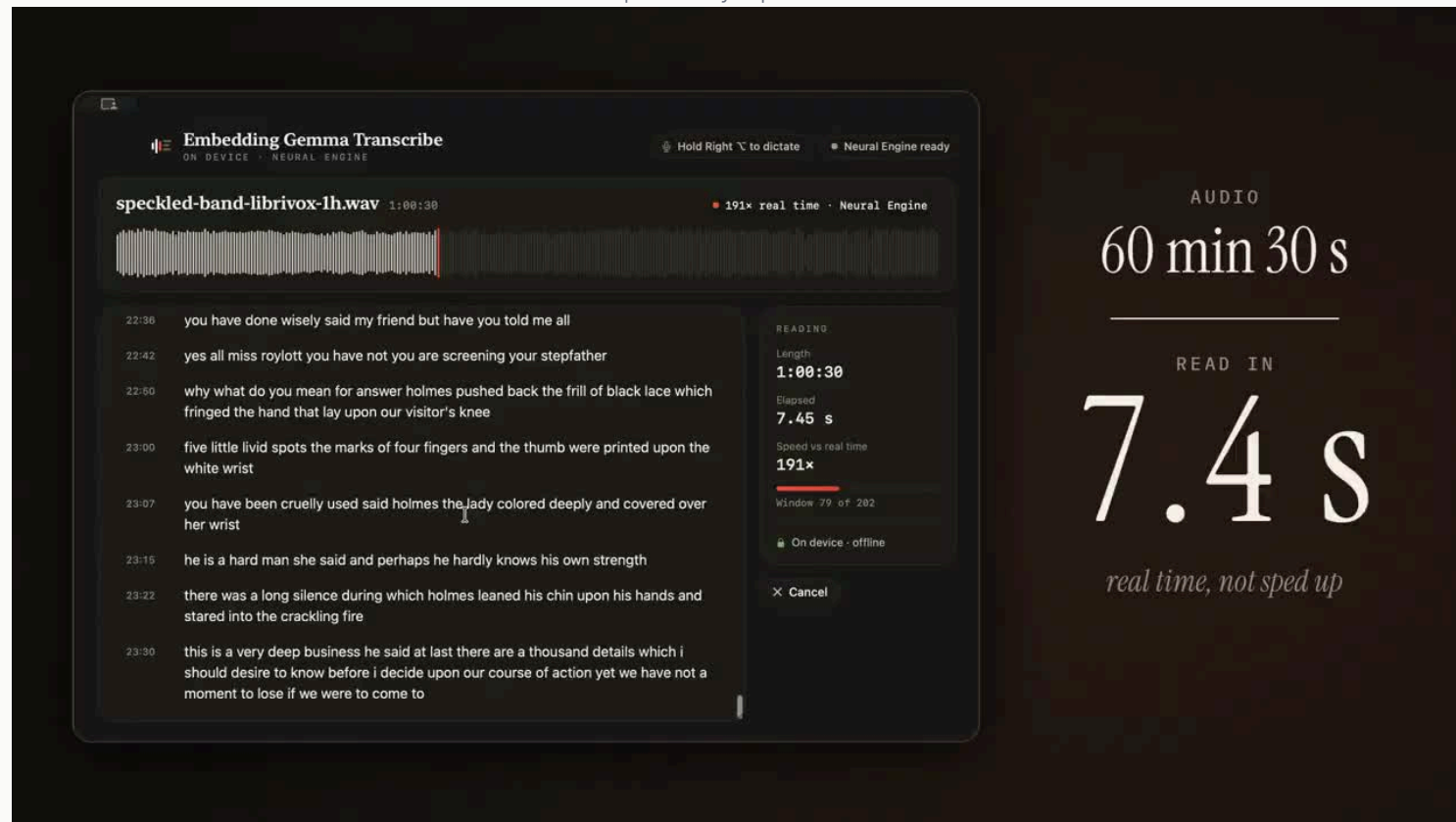
Try this with our slides: query → ranked excerpts → lecture + slide link.

[Open the course-search lab](#)

Retrieval returns evidence you can inspect. A generated answer would be an extra step.

Can an embedding model help transcribe speech?

AUTHOR DEMO · 12 October 2026 · benchmark not independently reproduced



AUTHOR-REPORTED

~13 s

to process 1 hour of audio
base M3 MacBook Air

5.18%

word error rate
LibriSpeech test-clean

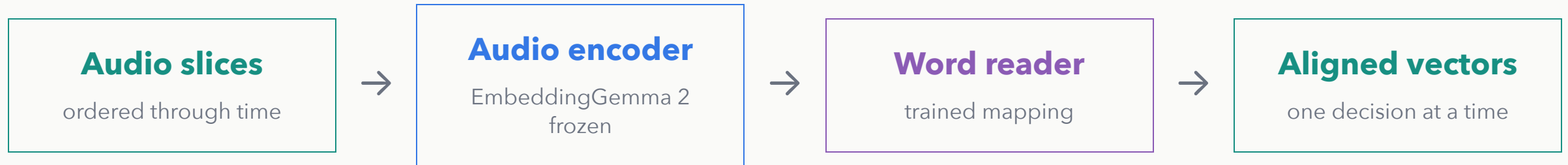
EmbeddingGemma stays frozen.
A separate reader is trained.

[Watch the original X demo](#) · Shahir · 26 s

A shared space is useful, but a transcript still needs a sequence of word decisions.

Keep the audio order; learn how to read it

SOURCE-GROUNDED SCHEMATIC · internal layers and slice timing not disclosed



Audio slices

t_1 t_2 t_3 ...

Reader outputs

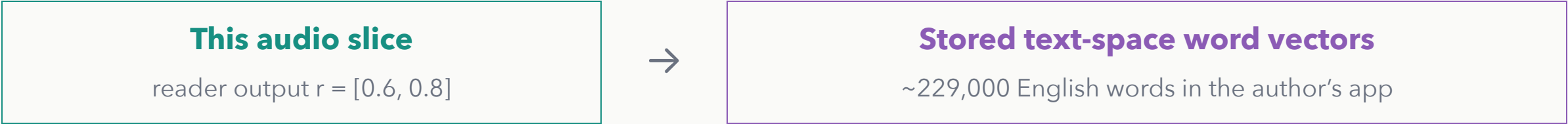
r_1 r_2 r_3 ...

A single pooled vector is useful for retrieval.
Exact words require decisions across time.

The author describes a reader over audio slices, not inversion of one whole-recording vector.

Compare a reader output with candidate words

ILLUSTRATIVE · 2D arithmetic, invented vectors; exact lookup metric not disclosed



Candidate	Toy word vector	Illustrative cosine $r \cdot e_w$
cat	$[0.8, 0.6]$	$0.6 \times 0.8 + 0.8 \times 0.6 = 0.96$
cap	$[1, 0]$	0.60
dog	$[-0.6, -0.8]$	-1.00

$\arg \max_w (r \cdot e_w) \rightarrow \text{cat} \rightarrow \text{assemble words over time} \rightarrow \text{punctuation}$

The comparison looks familiar. The learned reader makes it useful for recognizing words.

First build a small, labelled word-recognition task

PROPOSED TEACHING DATASET · not a reconstruction of the author's training corpus

INPUT

A short audio clip

mono · 16 kHz

one spoken command



TARGET

"yes"

integer label: 0

one label per clip

Speech Commands v0.02: use yes, no, up, down, left, right, stop, go.

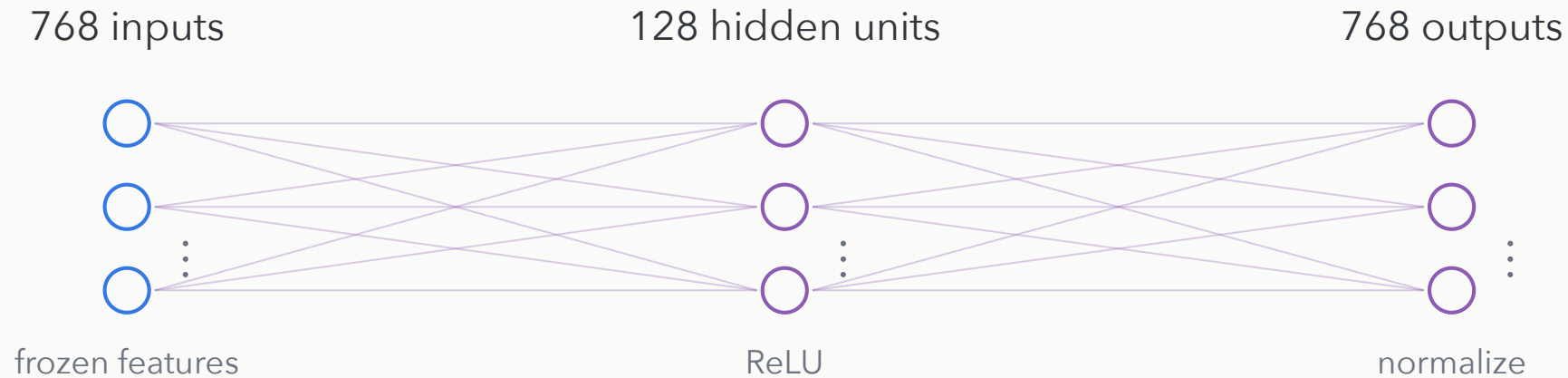
Use the official train / validation / test lists; keep speakers separate.

Cache an $N \times 768$ feature matrix and N integer labels for each split.

This is an eight-word teaching experiment. It is not yet full speech-to-text.

A 768-dimensional vector is 768 input features

PROPOSED TEACHING READER · 768 → 128 → 768; not the published app's architecture



$X: \mathbf{B} \times 768 \rightarrow \text{hidden: } \mathbf{B} \times 128 \rightarrow R: \mathbf{B} \times 768$

$R E^T / \tau \rightarrow \mathbf{B} \times 8 \text{ word scores}$

E stores eight frozen text vectors: 8×768 . Here $\tau = 0.1$ is a chosen hyperparameter.

Train the small mapping. Keep the embedding model and candidate word vectors fixed.

The correct word supplies the training target

ILLUSTRATIVE PROBABILITIES · chosen training objective for our small experiment

Audio says **"yes"** → target $y = 0$

Candidate word	yes	no	up
Illustrative softmax share	0.70	0.20	0.10

$$L = -\log p(\text{correct word}) = -\log 0.70 \approx \mathbf{0.357}$$

loss → gradients → reader weights

· frozen encoder and word bank stay fixed

Average this loss over the minibatch, update the reader, then evaluate on held-out speakers.

The training step fits on one slide

TEACHING CODE · cached real features required; model training results are not claimed

```
reader = nn.Sequential(nn.Linear(768, 128), nn.ReLU(),
                       nn.Linear(128, 768))
optimizer = torch.optim.AdamW(reader.parameters(), lr=1e-3)

for X, y in loader:
    # (B, 768), (B,)
    R = F.normalize(reader(X), dim=-1) # (B, 768)
    logits = R @ E.T / 0.1             # E: (8, 768); logits: (B, 8)
    loss = F.cross_entropy(logits, y)  # mean across the minibatch
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

X is a cached embedding batch. y is a batch of word labels. E is a frozen word-vector table.

Full transcription adds an alignment problem

PRACTICAL NEXT STEP · proposed route to ASR; author's exact training recipe is not public

INPUT

An entire utterance

many words, different durations



TARGET

“please turn left”

an ordered sequence of words

A practical dataset

LibriSpeech: paired English speech and transcripts.
Start with train-clean-100.

An extra design decision

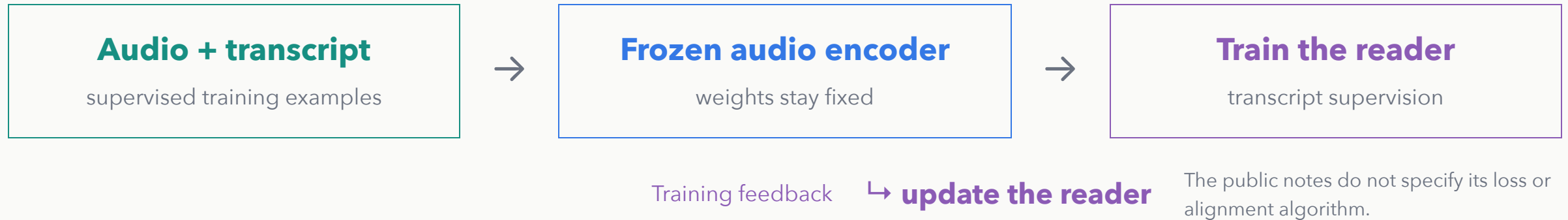
Keep time-indexed features.
Learn alignment, or obtain word timings before training.

Validate on dev-clean; reserve test-clean / test-other for final WER.
Keyword accuracy does not establish sentence transcription quality.

One pooled 768D vector per recording does not preserve a usable word-by-word timeline.

The embedding model is reused; the reader learns a new job

AUTHOR METHOD · high-level training view; no invented loss or reproduced benchmark



DIGUP

Encode files → store vectors
→ retrieve a source location

SPEECH DEMO

Audio features → trained reader
→ word sequence

What would you test next? **Names, accents, noise and long recordings.**

Shared representations enable new applications. The task still determines the readout and the evaluation.