

vibe coding-Andrej Karpathy ל-agentic engineering

ריאיון במה על Software 3.0, verifiability, אינטליגנציה "משוננת" (jagged) ומה עדיין
נשאר באחריות האדם כשה-agents כותבים את הקוד

Andrej Karpathy (מרואיין), מראיינת שלא הוצגה בשמה · AI Ascent (לפי הרקע על הבמה ואזכור בתמלול). הסרטון
הגיע מפרסום חוזר ב-X של Movez · פורסם ב-X ב-2026-06-08; תאריך האירוע המקורי לא ידוע

משך: כ-30 דקות · X

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זה ריאיון במה (fireside chat) של כ-30 דקות מאירוע AI Ascent. השם מופיע על המסך שמאחורי הדוברים, ובשיחה עצמה יש אזכור ל-AI Ascent של השנה הקודמת. המראיינת לא מציגה את עצמה בשמה. את המראיין היא פונה אליו בשם "Andre", ומציגה אותו כמי שהיה שותף להקמת OpenAI, הוביל את Autopilot ב-Tesla וטבע את המונח vibe coding. התיאור הזה מתאים ל-Andrej Karpathy. הכותרת והתיאור של הפרסום החוזר ב-X (למשל "90% of my coding routine is automated") לא מופיעים בתמלול, ולכן לא נלקחו מהם עובדות.

הטענה המרכזית: מאז דצמבר האחרון, agents לכתובת קוד עברו מ"עוזר שכותב קטעי קוד" למערכת שאפשר לסמוך עליה, ולכן צריך לחשוב בפרדיגמה חדשה (Software 3.0) ולא רק לעבוד מהר יותר בישנה. Karpathy מבחין בין vibe coding, שמרים את הרצפה לכולם, לבין agentic engineering, שמשמר את רף האיכות של תוכנה מקצועית. הוא גם טוען שהמודלים חזקים במיוחד בתחומים שאפשר לאמת (verifiable) ושמעבדות ה-AI התמקדו בהם. לכן האדם נשאר אחראי על ה-spec, על הטעם ועל ההבנה.

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות בשמונה סעיפים, ומה לא נאמר.
- 2. **נקודת המפנה של דצמבר:** למה Karpathy מרגיש "הכי מאחור אי פעם" כמתכנת.
- 3. **Software 3.0:** OpenClaw, Menugen, knowledge bases ומחשבים נירוניים.
- 4. **Verifiability ו-jagged intelligence:** למה המודלים מצטיינים בקוד ונכשלים בשאלת שטיפת המכונית.
- 5. **vibe coding מול agentic engineering:** גיוס, טעם, שיפוט ו-spec.
- 6. **עולם agent-native וחינוך:** תשתית ל-agents, "רוחות רפאים" ולמה ההבנה לא ניתנת למיקור חוץ.
- 7-8. **פעולות ומילון:** מה לעשות מחר בבוקר, שאלות פתוחות ומונחים.

הערות מקור המסמך הופק אוטומטית מהתמלול ומהפריימים של ההקלטה. בסרטון אין שקפים, יש רק צילומי במה. שמות ומונחים עלולים להשתבש בתמלול האוטומטי (ASR), למשל "lot code" במקום Claude Code ו-"ChachiPT" במקום ChatGPT, ובמקומות כאלה זה מצוין. הציטוטים מועתקים מילה במילה ונבדקו מול חותמת הזמן שלהם. לא נוסף מידע ממקורות חיצוניים.



הבמה ב-AI Ascent: המראיין (משמאל) והמראיינת (מימין) מול הקהל. זה הפריים היחיד שמזהה את הדוברים, כי בהקלטה אין שקפים

1. תקציר מנהלים

זו לא הרצאה מובנית אלא שיחה של שאלות ותשובות. עם זאת, אפשר לזהות בה קו מחשבה עקבי: הכלים השתנו מהותית, הפרדיגמה השתנתה, ומה שמבדיל מהנדס טוב הוא היכולת לתזמר agents "משוננים" בלי לוותר על איכות. אלה הטענות העיקריות:

- **דצמבר היה נקודת מפנה:** לדבריו, עם המודלים האחרונים הקוד "פשוט יצא תקין", הוא הפסיק לתקן, התחיל לסמוך על המערכת ועבר ל-vibe coding מלא [1:32].
- **תכנות הוא עכשיו "איזה טקסט להדביק ל-agent":** ההתקנה של OpenClaw היא קטע טקסט שמדביקים ל-agent במקום סקריפט bash מסובך [4:35–3:03].
- **חלק מהאפליקציות פשוט לא צריכות להתקיים:** MenuGen, האפליקציה שבנה, הפכה מיותרת כשראה שמספיק לתת ל-Gemini תמונה של תפריט ולבקש מ-Nano Banana לצייר עליו את המנות [6:06–4:35].
- **LLMs מאוטמטים את מה שאפשר לאמת:** מחשבים קלאסיים מאוטמטים את מה שאפשר להגדיר בקוד, LLMs את מה שאפשר לאמת. לכן היכולת "משוננות" ומושפעות ממה שהמעבדות בחרו להכניס לאימון [12:19–9:16].
- **Jaggedness היא סימן שצריך להישאר בלולאה:** Opus 4.7 מבצע refactor לבסיס קוד של 100 אלף שורות, ובכל זאת ממליץ ללכת ברגל לשטיפת מכוניות שנמצאת במרחק 50 מטר [10:48].
- **vibe coding ו-agentic engineering הם שני דברים שונים:** הראשון מרים את הרצפה, השני שומר על רף האיכות המקצועי. לדבריו מהנדסים טובים בזה משיגים הרבה יותר מ-[15:33–17:08] 10x.
- **תהליכי הגיוס מיושנים:** במקום חידות, הוא מציע פרויקט גדול כמו Twitter clone ל-agents, ואז לנסות לפרוץ אותו עם עשרה agents של [18:39] Codex.
- **האדם אחראי על ה-spec, הטעם וההבנה:** דוגמת הבאג ב-MenuGen, שבה זוהו משתמשים לפי email במקום user ID, וגם הציטוט "you can outsource your thinking, but you can't outsource your understanding" [20:11, 27:59].

מה לא נאמר

- **אין מספרים:** לא נמסרו מדדי פרודוקטיביות, עלויות או אחוז הקוד שנכתב על ידי agents. הנתון "90%" מהכיתוב ב-X לא מופיע בתמלול.
- **ה-workflow עצמו לא מוצג:** Karpathy לא מתאר את ה-setup שלו (כלים, הגדרות, skills), אלא רק ממליץ "להשקיע ב-setup".
- **התחום ה-verifiable הסודי:** הוא רומז שיש תחום RL בעל ערך שלא נוצל, ומסרב לחשוף אותו [14:00].
- **"הרבה יותר מ-10x זה רושם אישי":** נאמר במפורש "from my perspective", בלי נתונים.
- **אין התייחסות לאבטחה בפועל:** מעבר לכך שאסור להכניס vulnerabilities, לא נדונו פרקטיקות review, בדיקות או governance.

2. נקודת המפנה של דצמבר: "מעולם לא הרגשתי כל כך מאחור"

השיחה נפתחת בציוץ של Karpathy, שלפיו הוא "מעולם לא הרגיש כל כך מאחור כמתכנת". התחושה, לדבריו, היא שילוב של התלהבות ואי-נוחות. במשך כשנה הוא השתמש בכלים agentic (בתמלול "lot code", כנראה Claude Code). הכלים היו טובים בכתיבת קטעי קוד, אבל לפעמים טעו ודרשו תיקון.

השינוי הגיע בדצמבר: "December was this clear point where for me, I was on a break, so I had a bit more time." [1:32] האחרונה. מכאן הגיעה ההמלצה שחזר עליה ב-X: מי שהתרשם מ-AI דרך ChatGPT בשנה שעברה צריך להסתכל שוב, כי ה-workflow-ה-agentic "באמת התחיל לעבוד". התוצאה אצלו: תיקיית side projects "מלאה מאוד", ו-vibe coding כל הזמן.

3. Software 3.0: לתכנת עם טקסט, ואפליקציות שלא צריכות להתקיים

Karpathy חוזר על החלוקה המוכרת שלו: Software 1.0 הוא קוד מפורש, Software 2.0 הוא תכנות באמצעות datasets ואימון רשתות, ו-Software 3.0 הוא prompting. מודל שאומן על מספיק משימות הופך ל"מחשב שניתן לתכנות", וחלון ה-context הוא "הידית שלך על ה-interpreter".

הדוגמה הראשונה היא OpenClaw. במקום סקריפט התקנה שמתנפח כדי לתמוך בכל הפלטפורמות, ההתקנה היא קטע טקסט שמדביקים ל-agent. ה-agent בוחן את הסביבה, פועל בצורה אינטליגנטית ומדבג בתוך הלולאה.

— Andrej Karpathy [4:35] It's just like, what is the piece of text to copy paste to your agent? That's the programming paradigm now.

הדוגמה השנייה קיצונית יותר. Karpathy בנה את Menugen ב-vibe coding: מעלים תמונה של תפריט במסעדה, האפליקציה מבצעת OCR לשמות המנות, מפיקה לכל מנה תמונה במחולל תמונות, והכול רץ על Vercel. אחר כך הוא ראה את "גרסת Software 3.0": נותנים את התמונה ל-Gemini ומבקשים מ-Nano Banana לצייר את המנות ישירות על התפריט. התוצאה היא אותה תמונה, עם המנות מצוירות בפיקסלים. בלשונו: "all of my menu gen is spurious. It's working in the old paradigm." [6:06]

המסקנה הרחבה יותר: זו לא רק האצה של מה שכבר קיים, אלא עיבוד מידע כללי שאפשר עכשיו לאטמט. דוגמה לכך היא פרויקט LLM Knowledge Bases שלו, שבו ה-LLM "מקמפל מחדש" מסמכים לוויקי ארגוני או אישי. לדבריו, לא היה קוד שמסוגל לעשות את זה קודם. בהמשך הוא מותח את הקו עד "מחשבים נירוניים": מכשיר שמקבל וידאו או אודיו גולמיים ומשתמש ב-diffusion כדי לרנדר UI ייחודי לכל רגע. לדבריו, "the neural net becomes kind of like the host process" [7:43], "and the CPU has become kind of like the co-processor." [9:16]. ההתקדמות לשם הוא מגדיר כ-TBD.

4. Verifiability ו-jagged intelligence

Karpathy: "traditional computers can easily automate what you can specify in code, and kind of this latest round of LLMs can easily automate what you can verify in a certain sense." [9:16] המעבדות מאמנות במסגרת סביבות reinforcement learning ענקיות עם תגמולי אימות. לכן המודלים מגיעים לשיא במתמטיקה ובקוד, ו"מחוספסים בקצוות" מחוץ לתחומים האלה.

אבל זה לא רק עניין של verifiability, אלא גם של מה שהמעבדות בחרו לטפל בו. הדוגמה שלו: השיפור בשחמט בין GPT-3.5 ל-GPT-4 נבע, לדבריו, מכך שכמות גדולה של נתוני שחמט נכנסה ל-pre-training (בתמלול "pre-chaining"). הוא מציין שזה מידע ציבורי שראה באינטרנט. במילים אחרות, אנחנו "קצת נתונים לחסדי" ההחלטות של המעבדות.

— Andrej Karpathy [10:48] How is it possible that state-of-the-art Opus 4.7 will simultaneously refactor a hundred thousand line code base, or find zero-day vulnerabilities, and yet tells me to walk to this car wash. This is insane.

המסקנה המעשית: צריך לגלות באיזה "circuit" האפליקציה שלך נמצאת. "if you're in the circuits that were part of the RL, you fly." [12:19] וכדאי שישקול fine-tuning משלו. ליזמים הוא אומר שתחום שאפשר לאמת בו מאפשר לבנות סביבות RL ודוגמאות ולמשוך בידית בעצמך, גם אם המעבדות לא מתמקדות בו [14:00]. לדעתו, בסופו של דבר כמעט הכול ניתן לאימות, למשל כתיבה באמצעות "council of LLM judges". ההבדל הוא רק בין קל לקשה [15:33].

5. vibe coding מול agentic engineering: גיוס, טעם ו-spec

— Andrej Karpathy [15:33] I would say VibeCoding is about raising the floor for everyone in terms of what they can do in software. So the floor rises, everyone can VibeCode anything, and that's amazing, incredible. But then I would say agentic engineering is about preserving the quality bar of what existed before in professional software.

Karpathy קורא לזה "הנדסה" כי זו דיסציפלינה: לתאם agents "קוצניים", שגויים לפעמים ו-stochastic אבל חזקים מאוד, כדי להתקדם מהר בלי להוריד את רף האיכות. אסור להכניס vulnerabilities, ואתה עדיין אחראי לתוכנה שלך. לגבי התקרה: "10x is not the speed up you gain." [17:08] מי שטוב בזה מגיע להרבה יותר.

מה מבדיל משתמש AI-native ממשתמש בינוני? ניצול מלא של הכלים והשקעה ב-setup, בדיוק כמו שהשקיעו פעם ב-Vim או ב-VS Code. בגיוס, לדבריו, "most people are still not refactored their hiring process for" agentic engineer capability" [18:39]. במקום חידות, הוא מציע לתת פרויקט גדול, למשל Twitter clone ל-agents שמאובטח היטב ומדמה פעילות, ואז לנסות לפרוץ אותו עם "10-high x, 5.4, codecs". כנראה הכוונה לעשרה מופעים של Codex במודל 5.4 ברמת x-high, אבל השם לא ודאי בגלל ה-ASR.

מה נשבר: הבאג ב-Menugen

ההתחברות ל-Menugen היא עם חשבון Google, והקנייה של קרדיטים היא דרך Stripe. ה-agent קישר בין התשלום למשתמש לפי כתובת ה-email בשני השירותים, במקום לפי user ID קבוע. מי שהשתמש בכתובות שונות לא קיבל את הקרדיטים. "why would you use email addresses to try to cross-correlate the funds?" [20:11] לכן, לדבריו, האדם אחראי על ה-spec ועל התוכנית. "And I actually don't even like the plan mode." [20:11] הסיבה שלו: את ה-spec צריך לבנות יחד עם ה-agent, מפורט מאוד, "אולי בעצם ה-docs", ורק אחר כך לתת ל-agents לכתוב.

החלוקה שהוא מציע: את פרטי ה-API, כמו keepdims או keepdim, או dim, axis, reshape, permute או transpose, אפשר להשאיר ל"מתמחה" שיש לו זיכרון מצוין. את היסודות, כמו tensor, view ו-storage, והימנעות מהעתקת זיכרון מיותרת, אתה עדיין חייב להבין. תפקידך הוא "the aesthetics. the judgment, the taste, and a little bit of oversight." [18:39]

— Andrej Karpathy [21:42] I do think that when you actually look at the code, sometimes I get a little bit of a heart attack because it's not like super amazing code necessarily all the time, and it's very bloaty, and there's a lot of copy paste, and there's awkward abstractions that are brittle, and like it works, but it's just really gross

הדוגמה השנייה היא micro-GPT, ניסיון לפשט את האימון של LLM עד הסוף: "The models hate this. They can't do it." [21:42] זו לא מגבלה עקרונית, "המעבדות פשוט עוד לא עשו את זה".

6. עולם agent-native, "רוחות רפאים" וחינוך

על המאמר שלו בנושא "animals versus ghosts": אנחנו לא בונים בעלי חיים אלא "מזמנים רוחות רפאים". אלה מעגלי סימולציה סטטיסטיים, עם pre-training כמצע ו-RL שמוסיף "איברים" מעליו. הוא מודה שזה "קצת פילוסופיה", אבל יש לזה משמעות מעשית: "Like if you yell at them, they're not gonna work better or worse." [23:17] הגישה שהוא ממליץ עליה היא חשדנות וגילוי הדרגתי.

לגבי agents עם הרשאות אמיתיות: "הכול צריך להיכתב מחדש". הדבר שהכי מעצבן אותו הוא תיעוד שנכתב לבני אדם: "What is the thing I should copy paste to my agent?" [24:51] רוב הקושי ב-Menugen לא היה בקוד אלא ב-deployment ל-Vercel: חיבור שירותים, מסכי הגדרות ו-DNS. המבחן שהוא מציע לתשתית agent-native:

לתת ל-LLM את ה-prompt "build Menugen" ולקבל אפליקציה שרצה באינטרנט בלי לגעת בכלום. הכיוון שהוא רואה הוא ייצוג של אנשים וארגונים על ידי [26:22] "I'll have my agent talk to your agent".agents.

— Andrej Karpathy, מצטט ציוץ שלא צוין מי כתב אותו [27:59] It was something along the lines of you can outsource your thinking, but you can't outsource your understanding.

לדבריו, האדם הופך לצוואר בקבוק: הוא צריך לדעת מה בונים, למה זה שווה ואיך לכוון את ה-agents, ו"אי אפשר להיות במאי טוב" בלי הבנה. ה-knowledge bases משמשים אותו בדיוק לזה. כל מאמר שהוא קורא מצטרף לוויקי, והוא שואל עליו שאלות. "כל הטלה אחרת של מידע נותנת לי תובנה", והוא מתאר את זה כ-synthetic data generation מעל נתונים קבועים. בסוף השיחה הוא אומר שישמח לחזור בעוד כמה שנים ולבדוק אם "אוטמטו אותנו לגמרי מחוץ ללולאה".

7. מסקנות ופעולות

- לבחון מחדש את הכלים: מי שגיבש דעה על coding agents לפני דצמבר צריך לנסות שוב, לפי הטענה המרכזית בשיחה.
- לכתוב docs ל-agents: לכל ספרייה, כלי או תהליך התקנה פנימי כדאי לשאול "מה הטקסט שמדביקים ל-agent?", כמו בהתקנה של OpenClaw.
- לפני שבונים pipeline, לבדוק אם מודל אחד עושה את זה: לקח Menugen: ייתכן שחלק מהאפליקציה מיותר.
- למפות איפה המודל "טס" ואיפה הוא נאבק: לזהות אם המשימה נמצאת בתוך ה-circuits של ה-RL. אם לא, לשקול fine-tuning או סביבות אימות משלך.
- ה-spec הוא האחריות שלך: מזהים, מודל נתונים והחלטות ארכיטקטורה כמו user ID ייחודי לא משאירים ל-agent.
- לעדכן ראיונות עבודה: פרויקט גדול עם בדיקת עמידות של agents תוקפים במקום חידות.
- לבנות כלים להבנה, לא רק לביצוע: למשל wiki אישי שנבנה על ידי LLM מהחומרים שקוראים.

שאלה פתוחה	מה נאמר בשיחה
איזה תחום verifiable בעל ערך עדיין לא נוצל?	Karpathy רמז שיש תחום כזה וסירב לחשוף אותו [14:00].
האם טעם ושיפוט ישתפרו במודלים?	הוא מקווה שכן, ולדעתו אין לכך מניעה עקרונית. כרגע אין תגמול אסתטי ב-RL [21:42].
איך מודדים "הרבה יותר מ-10x"?	לא נמסרו מדדים. מדובר ברושם אישי.
מתי תשתית תהיה agent-native?	המבחן המוצע: "build Menugen" עם deployment מלא בלי מגע אדם. אין לוח זמנים.
האם ההבנה תעבור גם היא ל-agents?	הוא אומר שישמח לבדוק את זה בעוד כמה שנים [29:29].

8. מילון מונחים

מונח	פירוש
vibe coding	כתיבת תוכנה על ידי הנחיית מודל בלי לבדוק את הקוד לעומק. לפי Karpathy, מרים את הרצפה לכולם
agentic engineering	דיסציפלינה של תיאום agents כדי לעבוד מהר יותר בלי לוותר על רף האיכות של תוכנה מקצועית
Software 1.0 / 2.0 / 3.0	קוד מפורש / משקלות נלמדות מ-datasets / תכנות באמצעות context-ו prompt
context window	הקלט שהמודל רואה. "הידית" שלך על ה-LLM כמפרש
verifiability	האם אפשר לאמת אוטומטית את נכונות הפלט. מה שניתן לאימות מתאים ל-RL
jagged intelligence	יכולות "משוננות": מצינויות בתחום אחד ושגיאות בסיסיות בתחום סמוך
RL (reinforcement learning)	אימון באמצעות תגמולים בסביבות שבהן אפשר לאמת תוצאה
RL environment	סביבת אימון עם משימות ותגמולי אימות
fine-tuning	אימון נוסף של מודל קיים על נתונים או סביבות משלך
pre-training	שלב האימון הראשוני על כמויות גדולות של טקסט, "המצע הסטטיסטי"

פירוש	מונח
מסמך דרישות ותכנון מפורט, שלפי Karpathy הוא כמעט ה-docs עצמם	spec
מצב תכנון בכלי Karpathy coding agent. מעדיף spec מפורט יותר	plan mode
תשתית ותיקוד שנבנו קודם כול לצריכה על ידי agents	agent-native
חיישנים ומפעילים: פירוק עבודה לקלט מהעולם ולפעולה בעולם	sensors / actuators
הפרויקט של Karpathy: ויקי שנבנה על ידי LLM מתוך מסמכים	LLM Knowledge Bases
אפליקציה ש-Karpathy בנה ב-vibe coding: הוספת תמונות למנות בתפריט	Menugen
מחולל התמונות שהוזכר יחד עם Gemini	Nano Banana